エンタープライズ系ソフトウェア開発のための
見積技法及びプロジェクト管理支援に関する研究

2007 年 7 月

原 田　晃

エンタープライズ系ソフトウェア開発のための
見積技法及びプロジェクト管理支援に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2007 年 7 月

原 田　晃

内容梗概

工業製品の開発においては，品質，コスト，納期若しくは開発期間の所謂 QCD を計画どおり成し遂げることが重要であるが，ソフトウェアの場合，ハードウェアと違い，外から形が見えないために，QCD の達成が非常に難しいものとなっている．

本論文は，ソフトウェアの中でエンタープライズ系ソフトウェアと呼ばれている分野の開発に関する課題のうち，特に，見積精度，プロジェクト管理，開発要件の曖昧性に関連する課題の解決に取組んだものである．

エンタープライズ系ソフトウェアとは，医療，福祉，交通，電力，通信，製造，流通，金融，政府，自治体，教育等の分野での企業や行政機関の経営や活動を支援する経営情報システムのことである．エンタープライズ系ソフトウェアは大規模，複雑であるが，開発は企業や行政機関自身で行わず，専門の受託開発企業に委託されることが殆どである．したがって, 業務知識は開発をする受託開発企業側にはなく, 開発を委託する顧客側にある．このために，1)開発着手時に開発すべき要件が明確になっていない，2)開発着手後に仕様追加・変更が多発し，開発すべき要件が確定しない，3)開発規模，開発工数，開発期間の見積の精度が低い，4)プロジェクト管理が難しいという課題がある．

課題 1)については，開発すべき要件を確定するプロジェクトと，その要件に基づきソフトウェアを開発するプロジェクトに分割し, 要件を確定するプロジェクトが完了した後に, ソフトウェア開発のプロジェクトを開始するという方法を採用することで，解決を図ることが主流になりつつある．

本論文では，課題 2), 3), 4)の解決に取組んだ．

2)に対しては，仕様発散防止 QFD という，QFD(Quality Function Deployment)を応用した技法を開発した．QFD は品質機能展開と言い，顧客ニーズをプライオリティの高いものから実現するには，どのような機能を組込むべきかを，定量的，客観的に決定する技法である．QFD の定量的，客観的，重点化という特徴を，仕様の追加，変更が多く対策の急ぐ必要のある機能を抽出し, 仕様の早期確定に応用したものが仕様発散防止 QFD である．仕様発散防止 QFD を実際のプロジェクトに適用して評価したが, 仕様の発散を防止でき, 仕様の早期確定に効果の大きいことが確認できた．

3)に対しては，要素見積法と呼ぶ，開発の前段階や着手時に精度の高い見積が可能とな

る，ファンクションポイント法を応用した見積技法を提案した．ファンクションポイント法の標準である IFPUG 法は，トランザクションファンクションと呼ぶ処理系要素の種類が 3 種類と少なく，具体的な処理を想起するのが難しいために，見積に習熟した人が適用して初めて精度の高い見積が可能になる．要素見積法では，具体的な処理を想起するのが容易な，要素機能と呼ぶ 16 種類のトランザクションファンクションを導入することで IFPUG 法の課題の解決を図った．要素見積法を実際の事例に適用し評価したところ，開発の前段階や着手時に，見積に習熟していない人が見積を行っても精度の高い見積ができ，IFPUG 法の課題を解決していることを確認できた．

4)に対しては，WBS に基づくプロジェクト管理支援システム「プロナビ」を開発することで解決を図った．

エンタープライズ系ソフトウェアの開発プロジェクトは，作業や成果物が膨大な数になる，多数のプロジェクトメンバが複数の開発拠点に分散しておりプロジェクトの計画や状況，成果物等の情報共有が難しい，作業や成果物作成にあたって設計標準等の利用する資料も膨大であり適したものを捜す負荷が大きいという課題があり，プロジェクト管理を人手で実施するのでは非常に難しい．この解決のために，プロナビでは，工程，作業の階層化，そして成果物，参考資料等の情報の関連付けが重要であると考え，WBS モデルを用いたプロジェクト管理支援システム「プロナビ」を開発した．プロナビでは，WBS により，それらの情報及び情報の実体である電子ファイルを相互に関連付けし，一元管理することで情報の参照や利用を容易にし，1)プロジェクト計画時の工程，作業，成果物の明確化，2)プロジェクト進捗状況の把握，3)プロジェクトメンバ間での成果物の共有，4)規則，標準，ワークシート等の組織に蓄積された知識の活用，5)開発プロセス及び作業の標準化を実現した．

本論文では，第 1 章で研究の目的と概要を，第 2 章でエンタープライズ系ソフトウェアの特徴，課題及び関連技術・研究を，第 3 章で要素見積法を，第 4 章でプロナビを，第 5 章ではプロナビに追加した進捗情報の自動収集機能を，第 6 章で仕様発散防止 QFD を，第 7 章でまとめと今後の研究方針を説明している．

研究業績一覧

(1)学術論文誌

[1-1] 原田晃，幕田行雄，石川貞裕，大野治，楠本真二，井上克郎,“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，vol.J89-D, no.4, pp.755-766, April, 2006.

[1-2] 原田晃，粟根達志，伊野谷祐二，大里立夫，大野治，松下誠，楠本真二，井上克郎,“WBS に基づくプロジェクト管理システムの実現,”SEC journal, no.9, pp.10-17, Feb. 2007.

(2)国際学会（査読あり）

[2-1] Akira Harada, Osamu Ohno, kouji Uehara, Kouichi Fujii, Hikozo Komuro, Yuji Inoya, Chiaki Hirai, Katsumichi Yasuda, "Project management with "PRO-NAVI"," Proceedings of the International Conference on Project Management ProMAC2002, pp.73-79, July 2002.

[2-2] Akira Harada, Osamu Ohno, kouji Uehara, Hikozo Komuro, Makoto Kurashige, Katsumichi Yasuda, " Specific instability prevention QFD: A Practical application of QFD for business custom software project," Proceedings of the International Conference on Project Management ProMAC2002, pp.319-327, July 2002.

[2-3] Hikozo Komuro, Osamu Ohno, Yukari Furuhata, Yukio Makuta, Akira Harada, Yutaka Kudo, Katsumichi Yasuda, "A project management tool utilizing review reports for software development," Proceedings of the International Conference on Project Management ProMAC2002, pp.171-176, July 2002.

[2-4] Hikozo Komuro, Osamu Ohno, Yukari Furuhata, Katsumichi Yasuda, Akira Harada, Yukio Makuta, " Improving an accuracy of estimation in a software development with a specific program development automation," Proceedings of the International Conference on Project Management ProMAC2002, pp.461-469, July 2002.

[2-5] Hiroshi Isizaki, Satoshi Awane, Sadahiro Ishikawa, Akira Harada, "Measures for improving project time management with enhancement of project management tool PRO-NAVI," Proceedings of the International Conference on Project Management ProMAC2004, pp.203-208, Oct. 2004.

[2-6] Akira Harada, Satoshi Awane, Yuji Inoya, Osamu Ohno, Makoto Matsushita, Shinji Kusumoto, Katsuro Inoue, "Project management system based on Work-Breakdown-Structure process model," Proceedings of the International Software Process Workshop, SPW2005, pp.249-261, May 2005.

[2-7] Makoto Kurashige, Akira Harada, "A practical use of an estimate method at early stage of business application software development," Proceedings of the International Conference on Project Management ProMAC2006, pp.184-186, Mar. 2006.

(3)その他

[3-1] 今城哲二，吉野松樹，大坪稔房，原田晃，大野治，植村俊亮,“ オンライン業務プログラムの環境独立処理方式,” 情報処理学会第１３５ソフトウェア工学研究会, vol.2001, no.114, pp.33-40, (2001-11)

[3-2] 初田賢司，原田晃，大野治,“ ソフトウェア開発プロジェクトにおける見積技術,” プロジェクトマネジメント学会誌，vol.4, no.4, pp.14-18, Aug. 2002.

[3-3] Michio Tsuda, Sadahiro Ishikawa, Osamu Ohno, Akira Harada, Mayumi Takahashi, Shinji Kusumoto, Katsuro Inoue, "Effectiveness of an integrated CASE tool for productivity and quality of software development," IEICE TRANS. INF & SYST., vol.E89-D, no.4, pp.1470-1479, April 2006.

謝辞

本研究の全般に関し，常日頃より適切なご指導を賜りました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻 井上 克郎 教授に，心から深く感謝申し上げます．

本論文を執筆するにあたり，適切なご助言とご指導を頂きました，大阪大学大学院情報科学研究科情報システム工学専攻 菊野 亨 教授, 大阪大学大学院情報科学研究科コンピュータサイエンス専攻 楠本 真二 教授に心より感謝致します．

本研究を行うにあたり，適切なご助言やご指導を頂きました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松下 誠 准教授に心から感謝致します．

本研究を行うにあたり, ご助言やご指導を頂きました, 株式会社 日立製作所 大野 治氏, 石川 貞裕氏，幕田 行雄氏，粟根 達志氏，伊野谷 祐二氏，大里 立夫氏，小林 り恵氏，倉重 誠氏に感謝致します．

本研究を行うにあたり，ご助言やご指導を頂きました，株式会社 日立システムアンドサービス 加藤 允基氏に感謝致します．

本研究を行うにあたり，ご助言やご協力を頂きました，日本電子計算株式会社 内池 正名氏，澤 則夫氏，末延 龍雄氏，鈴木 一弘氏に感謝致します．

最後に，井上研究室の皆様のご助言，ご協力に御礼申し上げます．

目次

第１章　まえがき

1.1　本研究の目的

コンピュータやネットワーク等のハードウェアの高性能化や低価格化，Windows, UNIX や LINUX 等の高機能で操作性のよい基本ソフトウェアの進化，Java や.NET 等の高効率の開発環境の出現により，情報化社会の進展は益々，著しくなっている．

これらの基盤の上に，医療，福祉，交通，電力，通信，製造，流通，金融，政府，自治体，教育，その他殆どの分野に属する組織が高度な情報システムを構築し，日々の組織の活動に利用しており，今や情報システムなしでは，いかなる活動も考えられない．

情報システムは，コンピュータやネットワーク等のハードウェアと，オペレーティングシステムやデータベース管理ソフトウェア等の基本ソフトウェアと，それらの上で動作するエンタープライズ系ソフトウェアと呼ぶソフトウェアから成り立つと考えてよい．

エンタープライズ系ソフトウェアは，それを利用する組織の事業や運営と密接に関係したソフトウェアであり，例えば，交通分野には，座席予約や車両の運行管理を行うものがある．

各分野とも顧客へのサービスが高度化し，また，組織間の競争が激しくなっており，高度な情報システムを構築し，活用することによって対応している．情報システムの高度化に伴い，エンタープライズ系ソフトウェアは高機能，大規模かつ複雑なものになっている．更に，開発にあたっては，短い納期と低いコストで高い品質を確保することが要求され,開発の難易度が高くなっている．

エンタープライズ系ソフトウェアの開発を成功させるためには，設計や開発に関する高い技術力だけでなく，開発規模やコストを正確に見積り，それに基づいて適切なプロジェクト計画を作成し，品質や進捗を管理し，不具合点が見つかれば修正をし，プロジェクトメンバ間での効果的なコミュニケーションを図り，計画どおりに開発が進むようにプロジェクト管理を行うことも必要である．

エンタープライズ系ソフトウェアの開発においては，約 40%が失敗すると言われている[31]．この原因は，開発規模やコストの見積の誤り，人手にのみ頼った設計書の作成，不適切な進捗管理，ステークホルダーやプロジェクト内の

コミュニケーションの不足，機能仕様の追加や変更の頻発，変更管理の不履行等，多岐にわたる．

筆者は，日立製作所及び日本電子計算株式会社においてエンタープライズ系ソフトウェア開発のためのソフトウェアエンジニアリングやプロジェクト管理に携わってきた．実際に成功したプロジェクト，失敗したプロジェクトを目の当たりにし，これらを分析すると，次の目標 1, 2, 3 の技術を実現することが，特に重要であると考えるに至った．本研究の目的は，この 3 つの目標の実現にある．

目標 1：早い段階での明確な根拠に基づいた精度の高い見積技法

目標 2：大規模なプロジェクトのプロジェクト管理を幅広く支援するシステム

目標 3：機能仕様の追加，変更の多発を防止し，仕様を早期に確定する技法

1.2 本研究の概要

本研究では，1.1 節で説明した 3 つの目標を達成するために，以下の 4 点について研究を行った．

(1)ファンクションポイント法を応用した早期見積技法

(2)WBS に基づくプロジェクト管理システム

(3)プロジェクト管理システム「プロナビ」の進捗管理機能強化

(4)QFD を応用した機能仕様の早期確定技法

(1)は目標 1，(2), (3)は目標 2，(4)は目標 3 の実現のための研究である．

(1)ファンクションポイント法を応用した早期見積技法

プロジェクトの開始前や初期段階での，ソフトウェアの開発費，開発工数，開発期間の見積は非常に重要であるが，そのためには，まず対象となるソフトウェアの開発規模を見積り，それに品質や性能等の要素を加味して予測することが多い[10],[12],[33],[45],[51],[54].

一方，ソフトウェアの開発規模をファンクションポイント（以後，FP という）という尺度によって計測する，ファンクションポイント法（以後，FP 法という）が標準になりつつある[2],[32],[44]．FP 法は，ユーザの視点から機能量を定量化するため，開発言語や実装方法に影響されない値を得ることができ，優れた見積技法であると考えられている．ところが，プロジェクトの開始前や初期段

階では，発注者から提案依頼書（RFP: Request For Proposal）が提示されるのみで，実装すべき機能仕様が詳細には定まっておらず，FP 法による規模見積を実施するには，いくつかの困難を伴う．この課題を改善したものとして NESMA 法[24]，ユースケースポイント法[38],[56]，電中研法[53]等いくつか考案されている．

本研究では，プロジェクトの開始前や初期段階での適用を目的とする，要素見積法と呼ぶ新しい見積技法を提案する.要素見積法では,画面上の 1 個の GUI ボタンや 1 回のファンクションキー押下で実行される，ユーザ視点での入出力の最小単位を要素機能と呼び，16 種類の要素機能を予め定義しておく．対象となるソフトウェアに，これらの要素機能が，それぞれいくつ含まれているかを計測してソフトウェアの規模を算出する．

(2)WBS に基づくプロジェクト管理システム

プロジェクト管理とは，プロジェクトの適切な計画を作成し，プロジェクトを計画通りに進めることである．プロジェクトは複数の作業と，その結果である複数の成果物から成り立っていると考えられるので,プロジェクト管理とは,それらの作業を効率よく遂行できるようにすることと，作業や成果物を管理することと考えることができる．

大規模なソフトウェア開発のプロジェクトでは，作業や成果物が膨大な数に渡るほか，多数のプロジェクトメンバが複数の開発拠点に分散しており，プロジェクトの進捗状況の把握，プロジェクトの計画や状況及び成果物等の情報共有が難しい.また,作業にあたって参考にする設計標準等の資料も膨大であり，適したものを捜す負荷も大きい．このため，プロジェクト管理を人手で実施するのでは非常に難しく，支援するシステが必要とされている．

プロジェクトの作業や成果物は，WBS(Work Breakdown Structure)[3],[16],[22],[47]モデルにより階層化され，管理される．即ち，WBS の要素毎に，スケジュールや作業者を設定し，進捗を管理する．また,プロジェクト管理を支援するシステムには Microsoft Project[55], ProcessDirector[46], PMOffice Enterprise[25]等が存在するが，上記のモデルに基づいたものが殆どである．

作業者がある作業を行おうとする場合，別の作業者が作成した成果物や設計標準等の資料を参考にすることが多い．また，管理者は，報告を受けて進捗状

況を把握するだけでなく，成果物の内容を直接，自らの目で確認することで把握したい場合がある．これを実現するには，WBS の要素毎に，スケジュールを設定し，進捗を管理するだけでなく，成果物の実体や参考資料を対応づけ，更に WBS の要素間の関係をつけ，WBS の要素をキーにして，スケジュール，進捗状況，関連する成果物や参考資料を容易に参照することのできる仕掛が必要である．

本研究では，上記に述べたような WBS モデルに基づいた作業，成果物，参考資料を管理するプロジェクト管理支援の方法の提案と，その方法に基づくプロジェクト管理システム「プロナビ」を開発する．

(3)プロジェクト管理システム「プロナビ」の進捗管理機能強化

プロナビを実際に利用したプロジェクト管理者に対し，プロナビの適用評価のためのアンケートやヒアリングを実施したところ，プロナビはプロジェクト管理を支援する有効なシステムであることが確認できた．一方，成果物作成の進捗管理では，個々の成果物の状況から人手で進捗を集計しなくてはならず負荷が大きいので,進捗情報を自動的に収集する機能が欲しいという要望がでた．

そこで，プロナビ標準 WBS のモデルに機能の階層を新設し，機能毎に進捗情報を自動収集することができるように機能強化を行う．これにより，成果物作成の進捗の実態に基づく，恣意性を排除した進捗情報をタイムリーかつ容易に把握できることになり，プロナビを更に有効なプロジェクト管理システムとすることができる．

(4)QFD を応用した機能仕様の早期確定技法

ソフトウェアの開発では，当初設定した予算や納期を守ることができない，テスト段階や顧客への納品後に品質上の重要な欠陥が発見される等の，いわゆる失敗プロジェクトが多い．これらの主な原因は，見積誤りによるコスト超過と不適切なプロジェクト管理の 2 つであるが，それは，次のようなソフトウェアの性質に起因するところが多い．即ち，ソフトウェアは無形物であるため，プロジェクト初期の段階では仕様を明確にしにくく，プロジェクトの進行とともに少しずつ明確にしていかなければならないという性質である.そのことが，見積誤りの発生や，プロジェクト管理を難しくしているのである．

上記課題を解決するには下記のような方策をとることが必要と考えられる．
(a)ソフトウェアの仕様や制約，プロジェクトの作業内容，顧客との作業分担等，コストを左右する部分の要件や前提条件を見積段階で明確にすること．
(b)その上で，これらの変化を常に追跡し，コストやスケジュールへの影響を把握し，その影響度に応じた適切な措置をとること．

追跡すべき項目として特に重要なのが，ソフトウェアの仕様の確定の状況である．仕様追加，変更が多発し仕様が確定しないと，開発作業量も増加し，コスト増や納期遅れを引き起こすことが多い．開発規模の実績が見積時の 2 倍以上に達することもあり，ソフトウェアの仕様の確定状況を管理することがソフトウェアの開発においては極めて重要である．

品質機能展開（QFD（Quality Function Deployment）以降，QFD と呼ぶ）は，顧客のニーズ（品質）を設計・製造等の技術的手段（機能）へ変換するためのプロセスを提供する技法である[4],[5]．QFD は，定量的な評価手段により決定過程を明確化・可視化することで，定型化による意思伝達手段や重点指向，網羅性を実現する．

本研究では，ソフトウェアの仕様の確定状況の管理に，QFD の定量化，可視化，重点指向という考え方を応用することを提案する．先ず，開発すべき機能毎に，影響範囲の大きさや難易度を数値化する．次に，定期的にその機能の仕様の確定状況を数値化し，数値の遷移の状況を評価することで仕様の確定を急がねばならない機能を特定する．これにより，重要な機能の仕様を早期に確定でき，プロジェクトの円滑な推進を期待できる．

1.3 本論文の構成

本論文では,第 2 章でエンタープライズ系ソフトウェアの開発に関する課題，技術・研究動向を，第 3 章でファンクションポイント法を応用した早期見積技法を，第 4 章で WBS に基づくプロジェクト管理システム「プロナビ」を，第 5 章でプロジェクト管理システム「プロナビ」の進捗管理機能強化を，第 6 章で QFD を応用した機能仕様の早期確定技法を，第 7 章で，これらのまとめと今後の研究方針を説明する．

第 2 章　エンタープライズ系ソフトウェア開発を取囲む環境と課題

2.1　緒言

本章では研究対象であるエンタープライズ系ソフトウェアの開発を取囲む課題，技術動向，研究動向について説明する．

2.2　エンタープライズ系ソフトウェアとは

ソフトウェアの分類方法は，いくつか知られている．Capers Jones は[32]で，次のような 6 種類に分類している．

(1)エンドユーザソフトウェア

職業としてプログラマまたはソフトウェア技術者でない人々が，個人的に作成し，利用するソフトウェアを指す．物理学の研究者，建築家，会計士等の専門知識を持ち，かつ，プログラミングのための技術的な能力を持ち合わせている人々が作成することが多いが，その規模は大きくない．

(2)経営情報システム

給与システム，会計システム，窓口及び後方支援の銀行業務システム，保険の苦情管理システム，航空座席予約システム等のビジネスや経営を支援するために企業が開発しているソフトウェアを指す.また,多くの行政機関も市町村，都道府県，国レベルでの課税関連ソフトウェア，社会保障福祉システム，運転免許及び自動車登録システムを開発しているが，これらも経営情報システムに含まれる．

経営情報システムは,メインフレームで稼動する大規模なソフトウェアから，パソコンで動作する小規模なソフトウェアまで存在し，規模，型とも広範囲に及んでいる．

(3)受託開発ソフトウェア

依頼主である企業や行政機関からの依頼に対して，受託開発企業が契約に基づいて開発する特定のソフトウェアを指す．

通常，規模の大きな経営情報システムの開発は，受託開発企業に委託されることが多く，この傾向は益々広がっている．

(4)市販ソフトウェア

数百から数百万の顧客を対象とする大規模な市場向けに開発されたソフトウェアを指す．市販ソフトウェアには，Microsoft Word のようなワープロ，Lotus,

Excel のような表計算，Microsoft Project 等，多種多様のソフトウェアがある．

市販ソフトウェアの分野が多種多様なことは，UNIX, LINUX, DOS, Windows, MVS, VMS 等の多様なオペレーティングシステムと複数のハードウェアプラットフォームをサポートする必要があることに拠る．また，パソコンの能力が高くなるにしたがって，市販ソフトウェアの規模は大きくなっている．

(5)システムソフトウェア

物理的な装置を制御するソフトウェアを指す．典型的なシステムソフトウェアは，UNIX, LINUX, DOS, Windows, MVS, VMS 等のコンピュータのハードウェアを制御するオペレーティングシステムである．また，PBX 交換機，医療装置のソフトウェア，携帯電話，電子レンジや乾燥機等の家庭用電化製品用の組込みソフトウェアも，これに含まれる．

(6)軍需ソフトウェア

米国等，軍隊の存在する国に特有の分類であり，空軍，陸軍，海軍等の軍や国防総省（または，これに相当する組織）のために開発されるソフトウェアを指す．これらのソフトウェアの例としては，指揮・管制・通信システムや後方支援システムがある．

この 6 種類のソフトウェアの中で，受託開発ソフトウェアを除くものは，用途に基づいた分類によるものであるが，受託ソフトウェアは，その開発の形態によったものである．即ち，依頼者からの委託によって開発されるソフトウェアを指しており，開発されるソフトウェアは主に経営情報システムであると考えてよい．したがって，[32]の分類は，本質的には，エンドユーザソフトウェア，経営情報システム，市販ソフトウェア，システムソフトウェア，軍需ソフトウェアの 5 種類と言ってよい．

一方，SLCP-JCF98[40]では，ソフトウェアを次のように分類している．

(1)業務システム

(a)事務系アプリケーションソフトウェア

(b)制御系ソフトウェア

(c)エンジニアリング系ソフトウェア

(2)ソフトウェア製品

(a)ファームウェア

(b)基本ソフトウェア

(c)パッケージソフトウェア

(d)ソフトウェア部品

Capers Jones と SLCP-JCF98 の分類の対応を表 2.1 に示す.

表 2.1　ソフトウェアの分類

Capers Jones による分類	SLCP-JCF98 による分類
エンドユーザソフトウェア	－
経営情報システム	事務系アプリケーションソフトウェア エンジニアリング系ソフトウェア
市販ソフトウェア	パッケージソフトウェア ソフトウェア部品
システムソフトウェア	制御系ソフトウェア ファームウェア 基本ソフトウェア
軍需ソフトウェア	－

エンジニアリング系ソフトウェアには CAD(Computer Aided Design)，即ち，コンピュータを用いて設計するための支援ソフトウェアと，製造業での設計図や部品を管理するシステム等があるが，ここでは後者を指しており，CAD はパッケージソフトウェアに含まれている.

また，ソフトウェア部品は，再利用部品，フレームワーク，テンプレート等を指す．これらを，業務システムを開発する時に利用することで，高い品質と生産性を確保することができる.

本研究で扱うエンタープライズ系ソフトウェアとは，[32]の分類での経営情報システムを指す.

2.3　エンタープライズ系ソフトウェアの特徴

既に説明したように，エンタープライズ系ソフトウェアは，企業や行政機関のビジネスや経営を支援する経営情報システムのことである．企業や行政機関には，医療，福祉，交通，電力，通信，製造，流通，金融，政府，自治体，教育等の非常に多くの分野が存在する.

これらの分野によって，必要とする経営情報システムは異なる．例えば，金融分野では勘定系オンラインシステムが必要であり，交通分野では座席予約システムが必要となる．

また，同じ分野の企業や行政機関の経営情報システムであっても，組織によって，微細な部分は異なることが多いし，それが組織の特徴となる．例えば，A 銀行と B 銀行では，同じ勘定系オンラインシステムでも，バックアップシステムの方式等，微細なところは異なるであろう．C 鉄道と D 鉄道では，同じ座席予約システムでも，座席番号の付け方や予約開始日等，微細なところは異なるであろう．

更に，一つの組織内でも，生産管理システム，社員の給与計算システムや旅費生産システム等，複数の経営情報システムが必要となる．

このように，エンタープライズ系ソフトウェアは，多くの用途のものが存在するが，その特徴を次のように纏めることができる．

(1)多種多様である．

上に説明したように，分野，組織毎に異なり，また，一つの組織内でも複数の経営情報システムが存在する．

(2)大規模である．

組織の事業，経営，活動を支援するシステムであるために，組織のプロセスそのものがシステム化されていると言える．また，画面，帳票等，人間とのインタフェースが多い．これらをソフトウェアで実現するために，規模が非常に大きくなることが多い． したがって，ソフトウェア技術者等，プロジェクトに参加するメンバも膨大になる．

(3)個別開発である．

例えば，同じ座席予約システムでも，鉄道会社によって，微細なところに差異があるので，汎用的な座席予約システムを開発し，これを異なる鉄道会社に適用することが難しく，個別開発となることが多い．

(4)受託開発が多い．

いったん開発されると，途中で規模の小さな改善のような開発はあるが，長期間にわたって利用されることが多く,新規に開発することは 5 年とか 10 年に 1 回というような頻度である．そのために，自社内にソフトウェア技術者を抱えておくことが難しい．また，新規開発では規模が大きく，開発期間が長くな

り，開発の難易度が高いことが多い．これらの理由により，自社でエンタープライズ系ソフトウェアを開発しないで，受託開発企業に開発を委託することが多い．

(5)プロジェクト型の開発である．

エンタープライズ系ソフトウェアの開発では，開発の開始と終了が明確である.オペレーティングシステムのようなシステムソフトウェアでは,いったん，リリースをしても，引き続きバージョンアップが行われ，同じ技術者が引き続き,開発を担当することが多い.一方,エンタープライズ系ソフトウェアでは，開発の開始時にプロジェクトが組織され，開発が完了して本稼動ができると，プロジェクトは解散される．したがって，同じメンバによるプロジェクトは殆どなく，プロジェクト内のコミュニケーションが重要となる．

(6)業務知識は顧客側にある．

エンタープライズ系ソフトウェアは企業，行政機関の各組織のプロセスそのものをシステム化したものであるので，業務知識に関しては顧客が深く持っており，ソフトウェア技術者は必ずしも詳しくはない．したがって，依頼主である顧客と一緒になって，業務仕様を確定する必要がある．ところが，顧客組織内には，経営者，システム部門，システムを利用するエンドユーザ等，ステークホルダーが沢山おり，業務仕様の確定が難しく，時間がかかる．

2.4　エンタープライズ系ソフトウェア開発に関する課題

エンタープライズ系ソフトウェアの開発においては，約 40%が失敗すると言われている[30],[31]．この原因は，エンタープライズ系ソフトウェア自身の特徴に内在していると言えるが，次のような課題があると考えられる．

課題 1：開発委託時に開発すべきソフトウェアの機能仕様が曖昧である．

エンタープライズ系ソフトウェアは，それを必要とする組織自身で開発しないで，受託開発企業に開発を委託することが多い．今後，便宜上，開発を委託する組織を購入者，開発を受託する受託開発企業を供給者と呼ぶ．このような場合，購入者は新しく開発するエンタープライズ系ソフトウェアの企画を行った後，開発するソフトウェアの要件を記載した提案依頼書（RFP: Request For Proposal）を作成し，複数の供給者に提案書の提出を依頼する．供給者は，その要件を満たすソフトウェアのアーキテクチャ，開発するためのコスト，期間

等を記載する提案書を作成し，購入者に提出する．購入者は，複数の供給者から提出された提案書の内容，供給者の能力を評価して，供給者を決定し，開発を委託する．

委託を受けた供給者はプロジェクトを組織し,プロジェクト計画を作成して,開発を進めていくことになる．ところが，開発の拠り所とする RFP には，開発するソフトウェアが備えるべき機能仕様が正確に記載されていないことが多い．

これは，次の理由にあると考えられる．

(a)購入者は，ソフトウェア開発に習熟していないために，どのレベルまで詳細に書くべきかがわからない．

(b)購入者の組織内には，経営者，システム部門，ソフトウェアを実際に利用するエンドユーザ部門等，ステークホルダーが沢山いる．ステークホルダーの要望や意見を RFP に反映する必要があるが，纏めるのが難しく，未完成ともいえる RFP を提出することが多い．

課題 2：開発規模，開発コスト，開発期間の見積精度が低い．

購入者,供給者の双方にとって,開発コスト,開発期間の見積は重要である．この見積の誤差が大きい場合，購入者にとっては，予算の超過や，新システムの予定した稼働日を守れないことになり，経営的な影響が大きい．供給者にとっても，開発コスト，開発期間の見積の誤差が大きいと，いわゆる赤字プロジェクトになり，やはり経営的な影響が大きい．したがって，開発コスト，開発期間を算出する基となる開発規模の見積が重要となる．

開発規模の見積は，RFP を基に行う．しかし，課題 1 で説明したように RFP 自身の記述が曖昧であり，正確に見積ることが難しい．

また,RFP に実現して欲しい要件が正確に記載されていたとしても,それは,ユーザの観点から直接，必要となる機能が記載されているのみで，その機能を実現するために必要となる 2 次的な機能については記載されていず，精度の高い見積を行うことは難しい場合が多い．

更に，見積技法にも課題がある．従来は，プログラムのコード行数（SLOC: Software Lines Of Codes）により開発規模を見積っていた．しかし，コード行数を尺度とする場合，開発言語に依存，開発者の技術レベルに依存，再利用可能プログラムの行数の扱い方の不統一等の課題が多い．最近では，ソフトウェア

の機能量をファンクションポイントという尺度によって定量的に計測する，ファンクションポイント法が標準になりつつある[32]．FP 法は，ユーザの視点から機能量を定量化するため，開発言語や実装方法に影響されない値を得ることができ，優れた見積技法であるが，習熟者でないと適用が難しい．

課題 3：機能仕様の確定に時間がかかる．

課題 1 で説明したように，いったんプロジェクトが開始されても，RFP に記載されている要件が曖昧なために，プロジェクトを進める中で，機能仕様を確定していく必要がある．また，RFP に正確に書いてある場合でも，プロジェクトを進めるなかで，機能仕様の変更や追加が発生することが多い．

このような場合，購入者側にはステークホルダーが多いために，要望や意見を纏めることが難しく，また，せっかく開発するシステムを少しでも良くしたいという気持が強いために，機能仕様の確定に時間がかかる．

この結果，仕様追加や変更の多発，開発規模の増大，開発コストの予算超過，進捗の遅延が発生し，プロジェクトの失敗の原因となることが多い．

課題 4：プロジェクト管理が難しい．

エンタープライズ系ソフトウェアの開発では，開発規模が大きく，作業や成果物が膨大な数に渡るほか，多数のプロジェクトメンバが複数の開発拠点に分散しており，大規模プロジェクトになることが多い．その結果，次のようなことが発生し，プロジェクト管理が難しくなる．

(a)作業，成果物，プロジェクトメンバが膨大になり，これらの整合性のとれたプロジェクト計画を作成することが難しい．

(b)プロジェクトの計画，状況，成果物等情報を，プロジェクトメンバ間で共有するのが難しい．

(c)作業や成果物が膨大な量になり，かつプロジェクトメンバが多いために，進捗状況の把握や，成果物の管理が難しい．

(d)作業の遂行，成果物の作成にあたって，関連する成果物や設計標準等の利用する資料が膨大であり，適したものを捜す負荷が大きい．

本研究の目的は,上記の 4 つの課題のうち，課題 2, 3, 4 の解決にある．1.1 節

で 3 つの目標を説明したが，目標 1 は課題 2 の解決を，目標 2 は課題 4 の解決を，目標 3 は課題 3 の解決を図るものである．課題 1 については，開発すべき要件を確定するプロジェクトと，その要件に基づきソフトウェアを開発するプロジェクトに分割し，要件を確定するプロジェクトが完了した後に，ソフトウェア開発のプロジェクトを開始するという方法を採用することで，解決を図ることが主流になりつつある．また，2.5.3(3)で説明する SEC の活動が抜本的な解決になるであろう．

2.5 エンタープライズ系ソフトウェアを取囲む技術動向，研究動向

エンタープライズ系ソフトウェアに関しては，これまで説明してきたような特徴や課題があり，それらの解決や改善を目的として，技術開発や研究が幅広く行われている．

2.5.1 見積技法

見積には，開発規模を見積ることの他，開発工数，開発期間や開発コストの見積がある．

(1)ファンクションポイント法

開発規模は，ソフトウェアの業界が誕生して以来 50 年間，プログラムのコード行数（SLOC: Software Lines Of Codes）で測定されてきた．しかし，コード行数を尺度とする場合の問題を，Capers Jones が 1978 年の IBM Systems journal に掲載した論文[28]で，初めて明確にした．

コード行数の問題点は，同じ Capers Jones の著書[32]で次のように整理されている．

(a)コード行数についての標準的定義が存在しない．

行の数え方に，物理行と論理行がある．物理行の終わりは ENTER キーである．一方，論理行の終わりはセミコロン，コロン，ピリオド等の区切り記号である.物理行で数えるか,論理行で数えるかで,算出結果に大きな差異が出る．

COBOL のような言語では，1 つの条件節が複数の物理行にわたる場合があり，物理行で測定した場合，論理行に比べ約 2 倍，規模が大きくなることがある．一方，Basic のような言語では，物理的な 1 行に複数の論理行が許され，論理行で測定した場合，物理行に比べ約 5 倍，規模が大きくなることがある．

(b)様々な種類の行が存在する[29].

殆どのプログラミング言語は，実行文，データ定義，コメント，空白行の 4 種類の行から成っている．この 4 種類の行のうち，どれを測定の対象にすべきかの標準が存在しない．

(c)再利用コードの測定が不正確である．

プログラマは，開発すべきプログラムの約 20%から 30%を，開発済みのプログラムからコピーし再利用している．再利用されたコードについて，1)再利用されたコードを出現ごとに測定する，2)再利用されたコードを一度だけ測定する，3)再利用コードは測定しない，の 3 種類の考え方が存在するが，いずれにするかの標準が存在しない．

(d)プログラマの技術レベル，プログラミングスタイルによる規模の差異が大きい．

同一の仕様を実現するために開発したプログラムのコード行数に，大きな差が生じることがある．これはプログラマの技術レベルやプログラミングスタイル，仕様の解釈の仕方に拠るものである．

(e)高水準言語による見かけ上の生産性低下が発生する．

アセンブラのような低水準言語から COBOL, Ada, VisualBasic のように高水準言語になるにしたがって，同じ仕様を実現するためのプログラムのコード行数は少なくなる傾向にある．一方，プログラムを完成させるためには，要件定義，アーキテクチャ設計等のためのコストの比率が大きく，全体の費用はコード行数が少なくなるほどは小さくならない．この結果，コード 1 行当りの費用 は高水準言語の方が高くなり，見かけ上の生産性は低下する．

これらのコード行数を尺度とする場合の課題を解決するものとして，ソフトウェアの機能量をファンクションポイントという尺度によって定量的に計測するファンクションポイント法が普及してきており，特にエンタープライズ系ソフトウェアでは FP 法が標準になりつつある[1],[15],[32],[33], [37],[39],[42],[50].

FP 法は 1979 年に Albrecht によって提案された[2]．これはユーザの視点から機能量を定量化するため，開発言語や実装方法に影響されない値を得ることができ，優れた見積技法であると考えられている．

企画段階や開発の前段階での規模見積が必要である．ところが，この段階では，実装すべき機能仕様が詳細には定まっておらず，FP 法による規模見積を実

施するには，いくつかの困難を伴う．この課題を改善するものとして NESMA 法[24]，ユースケースポイント法[38],[56]，電中研法[53]等いくつか考案されている．

(2)COCOMO, COCOMOⅡ

開発規模から開発工数,開発期間を算出する技法として,COCOMO, COCOMOⅡがある．

(a)COCOMO(COnstructed COst MOdel)

COCOMOは1981年にB. W. Boehmにより提案された工数見積技法である[8]. ソフトウェアのコード行数(SLOC)で計測した開発規模に，コスト誘因と呼ぶシステムの開発特性を反映した15個の工数変動要因や,開発モードと呼ぶプロジェクトの開発形態を加味して，開発工数と開発期間を算出するものである．

コスト誘因は，プロジェクトの開発環境の違いが開発工数に与える影響を考慮したものである．表2.2に示す15個の項目についてレベルを評価し，0.75～1.66の値を設定する．平均的なレベルの場合，1.00となる．

表 2.2　コスト誘因

属性	コスト誘因
ソフトウェア製品	ソフトウェア信頼性要求
	データベースの規模
	製品の複雑性
計算機	実行時間の制約
	主記憶の制約
	OSやハードウェアの変更頻度
	コンピュータのターンアラウンド時間
開発要員	分析者の能力
	アプリケーションの経験
	プログラマの能力
	OSやハードウェアの経験
	プログラミング言語の経験
プロジェクト	近代的プログラミング手法の使用
	ソフトウェアツールの使用
	開発スケジュール要求

また，プロジェクトの開発形態の違いが生産性の違いに表れると考えて開発モードを設定している．生産性の高い順に，組織モード，半組込モード，組込

モードの 3 種がある.

①組織モード：少人数での熟練度の高い開発形態.

②組込モード：大規模なプロジェクトで厳しい制約条件の下での開発形態.

③半組込モード：組織モードと組込モードの中間の開発形態.

これらの準備の下，COCOMO では，開発規模から開発工数と開発期間を，表 2.3 に示すように算出する.

表 2.3 COCOMO での開発工数，開発期間の算出式

開発モード	開発工数（人月）	開発期間（月）
組織モード	$MM=3.2(KSLOC)^{1.05}\cdot\Pi CD$	$TDEV=2.5(MM)^{0.38}$
半組込モード	$MM=3.0(KSLOC)^{1.12}\cdot\Pi CD$	$TDEV=2.5(MM)^{0.35}$
組込モード	$MM=2.8(KSLOC)^{1.20}\cdot\Pi CD$	$TDEV=2.5(MM)^{0.32}$

MM　：開発工数（人月）

KSLOC: ソースコード行数（1000 SLOC）

TDEV ：開発期間（月）

ΠCD ：15 個のコスト誘因(CD)の積.

(b)COCOMOⅡ

COCOMOⅡは 1997 年に公開された COCOMO の改訂版である[10],[12].

COCOMO はウォーターフォールモデルやインクリメンタルモデル等の，しっかりした開発プロセスによる大規模システムの開発が仮定されていた.一方，クライアント・サーバ･システム(CSS)の開発の場合，開発方法が多様化している，マルチベンダーによる製品の組合せが多様である，開発期間の短縮等のユーザの CSS に対する過信がある，ユーザの品質要求が多様化しているの問題があり，これらが生産性に影響を与える．それらに対応できるように COCOMO を改定したものが COCOMOⅡである.

COCOMO と同様に開発規模から次のように開発工数や開発期間を算出する.

・開発工数　$MM=2.45\times(KSLOC)^{B}\times\Pi EM_i$　(i=1～15)

・開発期間　$TDEV=2.66(MM)^{(0.33+0.2(B-1.01))}$

ここで B はスケールファクターと呼び，プロジェクトの特性の違いによってプロジェクトの生産性が，どう変化するかを説明する指標であり，$B=1.01+0.01(\Sigma W_i)$ で計算される.

W_i は表 2.4 に示す 5 個のスケール誘因の重みであり，COCOMOⅡの特徴である.

プロジェクトのレベルにしたがって評価され，0.00～6.07 の値をとる．

EMi は COCOMO のコスト誘因を拡張したものであり，表 2.5 に示す 17 個の項目についてレベルを評価し，0.75～1.67 の値を設定する．平均的なレベルの場合，1.00 となる．

表 2.4　スケール誘因

開発の先例性	関連するシステムの開発経験があるか，革新的な技術を必要としないか等の先例性の度合を表す．
開発の柔軟性	要求仕様や外部インタフェースに対する準拠の必要性の度合を表す．
アーキテクチャ／リスクの解決度	重要なモジュール・インタフェースや重要なリスクが解決されている度合を表す．
チームの凝集度	利害関係者間の文化の一貫性等，プロジェクトとしての纏りの程度を表す．
プロセスの成熟度	プロジェクトで使用されるプロセスの成熟度のレベルを表す．

表 2.5　COCOMOⅡでのコスト誘因

属性	コスト誘因
製品	ソフトウェア信頼性要求
	データベースの規模
	製品の複雑性
	ドキュメンテーションの適用性
プラットフォーム	実行時間の制約
	主記憶の制約
	プラットフォームの変更頻度
要員	分析者の能力
	アプリケーションの経験
	プログラマの能力
	プラットフォームの経験
	言語とツールの経験
	開発要員の継続性
プロジェクト	ソフトウェアツールの使用
	マルチサイト開発
	開発スケジュール要求

2.5.2 プロジェクト管理

(1)PMBOK

ソフトウェアの開発を成功させるためには，適切なプロジェクト管理を行うことが必要である．従来のプロジェクト管理は，QCD，即ち，品質，コスト，納期の 3 つに着目した管理手法であったと言える．

一方，最近では，顧客の満足する価値を届けるための開発計画を作成し，その計画を遂行することを目的としており，従来の手法と区別して，モダンプロジェクトマネジメント，若しくはプロジェクトマネジメントと呼ばれる．これは，1950 年代後半に米国国防総省が大規模プロジェクトを管理するために，マネジメント手法を体系化したのが始まりとされる．現在では，PMI(Project Management Institute)が PMBOK(Project Management Body Of Knowledge)と呼ぶプロジェクトマネジメントの遂行に必要な基本的な知識を汎用的な形で体系立てて整理している[3],[36]．PMBOK は 1984 年にプロトタイプ版，1987 年に第 1 版が発行され，2004 年に発行された第 3 版が最新のものである．

PMBOK は，QCD のみでなく，トータルプロジェクトマネジメントの観点にたち，スコープ，タイム，コスト，品質，組織，コミュニケーション，リスク，調達の 8 つの知識エリアと，8 つの知識エリアの統合の合計 9 つの知識エリアから構成されている．

次に 9 つの知識エリアの概要を説明する．

(a)スコープマネジメント

プロジェクトのスコープを明確にし，スコープの変更を管理するプロセスである．プロジェクトの目標を明確にし，それを実現するために必要な作業，成果物を全て洗い出し，WBS で記述する．

(b)タイムマネジメント

プロジェクトを所定の時期に完了させるために管理するプロセスである．スコープマネジメントで明確になった作業のスケジュールを作成し，スケジュール通りに進捗するようにコントロールする．このプロセスでガントチャートや PERT 図を作成する．

(c)コストマネジメント

プロジェクトを予算内で完了させるためのプロセスである．コストの計画，見積，予算化，コントロールのプロセスからなる．

(d)品質マネジメント

高品質なソフトウェアを実現するためのプロセスである．品質目標の設定，設定した品質目標を達成するために必要な具体的な活動の計画と遂行，結果の監視のプロセスからなる．

(e)人的資源マネジメント

プロジェクト内の役割・責任・報告ルートの明確化，プロジェクトチームの編成，プロジェクトメンバの育成，プロジェクトメンバが最大のパフォーマンスがあげられるようなマネジメントのプロセスからなる．

(f)コミュニケーションマネジメント

プロジェクト情報の生成，収集，配布，保管，検索，廃棄を実行するためのプロセスからなる．

(g)リスクマネジメント

プロジェクトのリスクを管理し，最小化するためのプロセスである．リスクの定量的・定性的な分析，リスクを減少させるための施策の策定，リスクの監視のプロセスからなる．

(h)調達マネジメント

プロジェクトを遂行する上で必要なプロダクト，サービスをプロジェクトの外部から調達するプロセスである．納入者選定，契約の管理のプロセスからなる．

(i)統合マネジメント

プロジェクト全体を計画通り進めていくためのプロセスである．プロジェクト憲章の作成，プロジェクトマネジメント計画書作成と遂行，プロジェクト状況の監視とコントロールのプロセスからなる．

(2)アーンドバリュー

スコープ，スケジュール，コストを統合し，プロジェクトの進捗を客観的に測定するプロジェクト管理手法である[18]．

スケジュール上の特定の期間で達成される予定の価値を計画価値(PV: Planned Value)，実際に達成された価値を達成価値 EV: Earned Value)，実際にかかったコストを実コスト(AC: Actual Cost)と呼ぶ．

具体的には，PV, EV は，次のように算出する．

・PV：プロジェクトでの作業全てを WBS に展開し，WBS の各構成要素を実施するために必要なコストを設定しておく．スケジュール上の特定期間で実行されることになっている WBS の構成要素のコストの合計が PV である．

・EV：スケジュール上の特定期間で実行された，WBS の構成要素のコストの合計が EV である．

通常，PV, EV, AC の関係は，図 2.1 に示すようになるが，進捗状況，コスト状況を同時に把握でき，優れた指標であると言えよう．

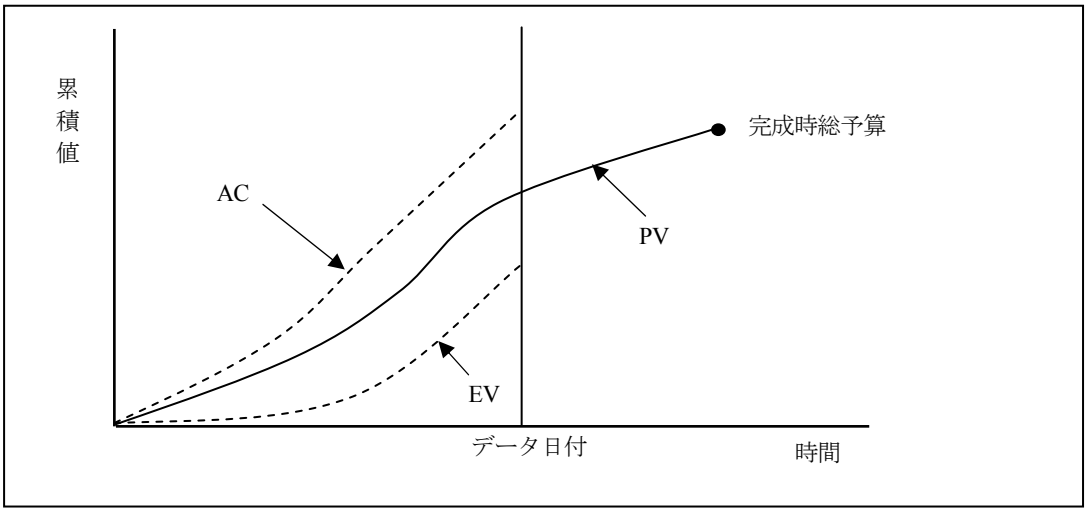

図 2.1　アーンドドバリューにおける各指標の関係

アーンドバリューは 1960 年代以前からも利用されていたが，1967 年に米国国防総省が，コスト／スケジュール管理システム基準(C/SCSC)を発表し，ある限度額以上の大規模なシステムの開発を委託する場合，委託業者に利用を義務づけた．以後，30 年間にわたり利用されてきた．この 30 年間の成果を踏まえ，1998 年に，米国国家基準／電子産業連盟 748 ガイド(ANSI/EIA748)として公布され，PMBOK にても，統合マネジメント，タイムマネジメント，コストマネジメントの技法として採用されている．

2.5.3　ソフトウェアエンジニアリング

ソフトウェアエンジニアリング(software engineering)という用語は，1968 年にドイツで開催された NATO（北大西洋条約機構）会議の表題として使われた

のが最初である.

IEEE Computer Society は，ソフトウェアエンジニアリングを,「ソフトウェアの開発，運用，保守を行うための工学的な方法である．工学的とは，方法論が確立しており，系統的かつ定量的な方法をいう.」と定義している[26].

(1)SWEBOK

IEEE によって「ソフトウェアエンジニアリング知識体系」が策定されたが，膨大な資料となっているために，それに対するガイドが，2004 年に SWEBOK 2004 として発行された[52].

SWEBOK には，次の 12 の知識領域が記述されている.

(a)ソフトウェア要求

対象とするソフトウェアに対する要求の，抽出，分析，仕様作成，妥当性確認に関する工学的な方法を扱う.

(b)ソフトウェア設計

ソフトウェア要求を実現するための，設計に関する工学的な方法を扱う.

(c)ソフトウェア構築

コーディング，検証，ユニットテスト，組合せテストに関する工学的な方法を扱う.

(d)ソフトウェアテスティング

ソフトウェアのテストに関する工学的な方法を扱う.

(e)ソフトウェア保守

ソフトウェアの開発が終了し運用に入った後の，ソフトウェアの改変，顧客に対する訓練，相談機能の運用等の，保守に関する工学的な方法を扱う.

(f)ソフトウェア構成管理

ソフトウェア及び付随する文書の,個々の構成要素の管理に関する工学的な方法を扱う.

(g)ソフトウェアエンジニアリングマネジメント

ソフトウェアの開発，保守，運用に対して，工学的な方法が適用されるように管理するための手法を扱う.

(h) ソフトウェアエンジニアリングプロセス

ソフトウェアの開発，保守，運用に対して，工学的な方法を適用していくためのプロセスを扱う.

(i) ソフトウェアエンジニアリングのためのツール及び手法

ソフトウェアの開発，保守，運用を支援するツール，技法を扱う.

(j)ソフトウェア品質

ソフトウェアの品質保証，検証，妥当性確認，レビュー，監査及び品質の計量化に関する工学的な手法を扱う.

(k) ソフトウェアエンジニアリングに関連する専門分野

ソフトウェアに対する工学的な方法と関連する科学分野,工学分野を特定する.その分野とは,コンピュータエンジニアリング,コンピュータサイエンス，マネジメント，数学，プロジェクトマネジメント，品質マネジメント，ソフトウェアエコノミクス，システムエンジニアリングである.

上記のように，SWEBOK は知識体系であるために抽象的であるが，ソフトウェアエンジニアリングの構成要素を網羅的に整理している．PMBOK を基にしてプロジェクトマネジメントが発展したように, SWEBOK を基にしてソフトウェアエンジニアリングの更なる発展が期待できるであろう.

(2)EASE プロジェクト

EASE(Empirical Approach to Software Engineering)プロジェクトは,文部科学省のリーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」の一環として 2003 年度より開始したプロジェクトで,ソフトウェア開発の分野におけるエンピリカルソフトウェア工学(Empirical Software Engineering)の確立を目指している.

エンピリカルソフトウェア工学とは，ソフトウェア開発プロジェクトのデータを常時，測定，定量化し，科学的根拠により分析・フィードバックするアプローチにより，生産性と品質の継続的な改善を支援する手法である.

EASE プロジェクトでは，プロジェクト管理者とプロジェクトマネジメントオフィスを支援することを目的として，EPM(Empirical Project Monitor)と呼ぶ観測型プロジェクト管理支援システム及び分析手法を開発した.

EPM 基本機能はデータ収集とデータ分析の支援システムであり，拡張機能は基本機能ではカバーしきれない，より高度な分析を行うのを支援するツール類である．EPM を実際のプロジェクトに適用し，データの収集と分析結果が得ら

れている[17],[43].

(a)EPM 基本機能

データを自動収集して，ソースコードの規模推移グラフ等の分析・評価・予測情報を出力する.

収集データには次のものがある.

・構成管理履歴情報

プログラムの変更・追加・削除やバージョン更新の日時等．これらの情報は構成管理ツール CVS(Concurrent Versions System)と連携して取得する.

・不具合履歴情報

・メール履歴情報

・ソースコード

(b)EPM 拡張機能

・協調フィルタリング機能

プロジェクトのデータに欠損値があっても，データ中の類似度の高い過去のデータから類推する機能.

・コードクローン検出ツール

ソースコードに存在するコードクローンを抽出するツール．コードクローンとは，同一若しくは類似したコードの断片で，多くは他のプログラムからコピーされたものである．コードクローン量を計測することで，ソフトウェア保守における潜在的リスクを把握できる.

・開発者間，プログラムモジュール間の類似分析

開発者間，プログラムモジュール間の類似性を分析，明示化．これにより，人員配置や作業割当の合理化，品質改善を行うことができる.

(3)SEC

SEC（ソフトウェアエンジニアリングセンタ）は，日本のソフトウェア産業において高品質なソフトウェア開発の実現を目的として，2004 年 10 月に，IPA（独立行政法人　情報処理推進機構）の中に設置された組織である．特にエンタープライズ系ソフトウェアと組込ソフトウェアの開発力強化に重心を置いている．本研究とはエンタープライズ系ソフトウェアの開発力強化と関係が深い

が，そこでは，開発プロセス共有化，定量データ分析，見積手法，要求工学・設計・開発技術，プロジェクト見える化，プロセス改善を研究しており，その成果が次の出版物となって発行されている．

①経営者が参画する要求品質の確保[41]

②IT ユーザとベンダのための定量的見積の勧め[27]

エンタープライズ系ソフトウェア開発に関して，2.4 節で，1) 開発委託時に開発すべきソフトウェアの機能仕様が曖昧である，2) 開発規模，開発コスト，開発期間の見積精度が低いことを説明したが，[27],[41]は，これらの改善を目的として発行された．[41]では，経営層が機能要件の確定に参加すべきであることが説かれている．[27]では，開発の前段階で定量的な見積を行うための要点が記載されている．

第 3 章　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

3.1　緒言

インターネットや Web の普及拡大に代表されるように情報化社会の進展は著しく，それに伴いエンタープライズ系ソフトウェアの開発においても大規模化，高機能化，短納期化，低コスト化の要求が急速に高まってきている．これを実現するためには，プロジェクトの開発計画を立て，開発計画どおりに開発が進むようにプロジェクト管理を行うことが必要である[23],[35],[48]．開発計画の中でも，開発費，開発工数，開発期間は，特に重要であると考えられており，これらを予測することを見積という．通常，開発費，開発工数と開発期間は，まず対象となるエンタープライズ系ソフトウェアの開発規模を見積り，それに品質や性能等の要素を加味して予測されることが多い[6],[7],[9],[11],[12],[13],[34],[45],[51],[54].

従来，ソフトウェアの規模はプログラムのコード行数（SLOC: Software Lines Of Codes）で測定されてきた．しかし，コード行数を尺度とする場合，2.5.1(1)で説明したように，開発言語に依存，開発者の技術レベルに依存，再利用可能プログラムのコード行数の扱い方の不統一等の課題が多く[28],[29],[32]，最近では，ソフトウェアの機能量をファンクションポイントという尺度によって定量的に計測する，ファンクションポイント法が普及してきており，特にエンタープライズ系ソフトウェアでは FP 法が標準になりつつある[32].

FP 法は 1979 年に Albrecht によって提案された[2]．これはユーザの視点から機能量を定量化するため，開発言語や実装方法に影響されない値を得ることができ，優れた見積技法であると考えられている．

FP 法によるエンタープライズ系ソフトウェアの規模見積の時点には，企画段階，開発の前段階，要求要件設計終了段階等があるが，プロジェクトの開発計画を立てるためには開発の前段階での規模見積が必要である．ところが，この段階では発注者から提案依頼書（RFP: Request For Proposal）が提示されるのみで実装すべき機能仕様が詳細には定まっておらず，FP 法による規模見積を実施するにはいくつかの困難を伴う．この課題を改善したものとして NESMA 法[24]，ユースケースポイント法[38],[56]，電中研法[53]等いくつか考案されている．

本論文では，開発の前段階でも FP を計測できる，要素見積法と呼ぶ見積技

法を提案する．また，この見積技法を実装した見積システム（AP-Estimate）について説明する．

画面上の 1 個の GUI ボタンや 1 回のファンクションキー押下で実行される，ユーザ視点での入出力の最小単位を要素機能と呼ぶ．要素見積法では，16 種類の要素機能を予め定義しておき，対象となるソフトウェアに，これらの要素機能が，それぞれいくつ含まれているかを計測して FP を算出するものである．これまで約 200 プロジェクトで，開発の前段階で AP-Estimate を用いた規模見積を実施しており，精度の高い結果を得ている．

以降，3.2 節で IFPUG 法の概要と課題を，3.3 節で要素見積法を，3.4 節で AP-Estimate の実現方式を，3.5 節で適用事例を，3.6 節で評価を行い，3.7 節では結論を述べる．

3.2 エンタープライズ系ソフトウェアの見積技法

3.2.1 エンタープライズ系ソフトウェアのライフサイクルと見積の時点

(1)ソフトウェアライフサイクル

図 3.1 に示すように，エンタープライズ系ソフトウェアの開発では，開発を委託する側と，委託を受けて開発する側の 2 つの組織が考えられる．[40]に倣って，開発を委託する側を購入者，開発を担う側を供給者と呼ぶことにする．購入者は新しく開発するエンタープライズ系ソフトウェアの企画を行った後，高品質のソフトウェアを低コストかつ短期間で開発できる供給者を選んで，開発を委託することが多い．

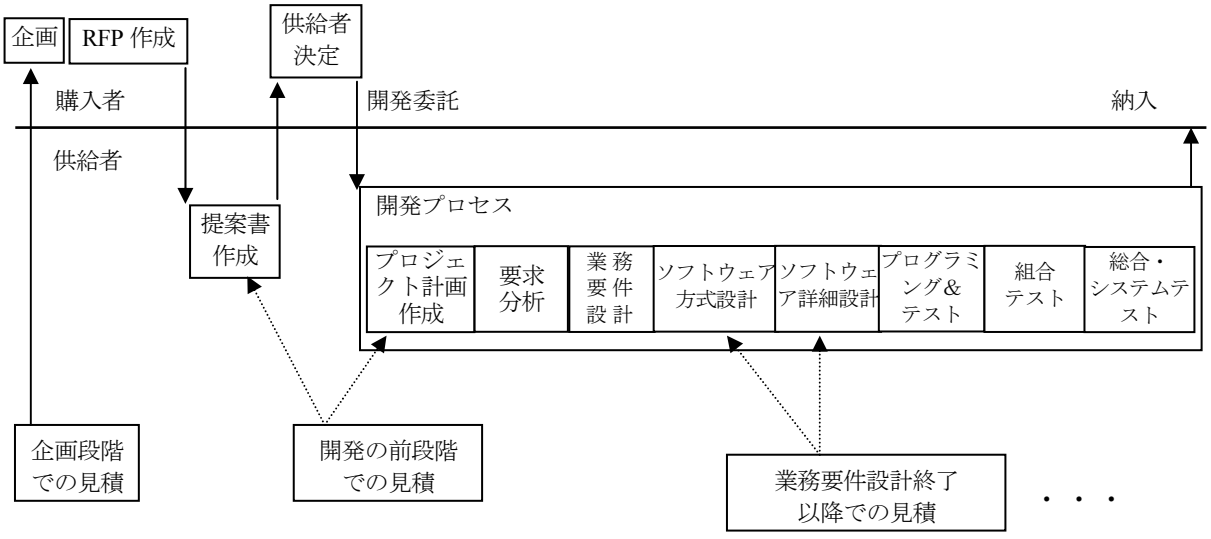

図 3.1 ソフトウェアライフサイクルと見積時点の関係

そのために，購入者は開発するソフトウェアの要件を記載した RFP を作成し，複数の供給者に提案書の提出を依頼する．供給者は，その要件を満たすソフトウェアのアーキテクチャ，開発するためのコスト，期間等を記載する提案書を作成し，購入者に提出する．購入者は，複数の供給者から提出された提案書の内容，供給者の能力を評価して，供給者を決定し，開発を委託する．
委託を受けた供給者はプロジェクトを組織し，プロジェクト計画を作成して，開発を進めていくことになる．

開発は，要求分析，業務要件設計，ソフトウェア方式設計，ソフトウェア詳細設計，プログラミング＆単体テスト，組合テスト，総合・システムテストというように複数の工程に分割され，この順序で行われていく．
このような開発プロセスは[40]で標準化されており，上記の 7 つの工程も標準化された開発プロセスと対応をつけることができる．

(2)見積の時点

ソフトウェアライフサイクルの様々な時点で見積を行うことは，企画の作成や，開発を進めていくうえでの改善に役立つが，特に，図 3.1 に示す次の 3 つの時点での見積が重要であると考えられる．

(a)企画段階での見積

予算化等を目的に開発費，開発期間を算出するために行う見積である．企画段階では，機能要件が明確になっていないことが多く，ここで見積る開発費は非常に粗い概算というべきものになる．

(b)開発の前段階での見積

図 3.1 に示すように，提案書の作成時点やプロジェクト計画作成時点等の開発の前段階で，開発費，開発工数，開発期間を算出するために行う見積である．購入者が作成した RFP に記載されている要件を基に見積を行う．RFP には，業務要件，ユーザ要件，機能要件が記載されている．業務要件とは，購入者である組織が，ソフトウェアを購入する理由や達成したいと望む目的を表す．ユーザ要件とは，ユーザがソフトウェアを使って実行できなければならない目標や仕事を表す．機能要件は，ユーザが仕事を遂行して業務要件を満たすために，ソフトウェアが備えていなければならない機能を表す．即ち，RFP には，実現して欲しいことは記載されているが，それをどのように実現するかは記載されていない．どのように実現するかは，開発プロセスの中で明確にされていく．

したがって，RFP には，ユーザの観点から直接，必要となる機能が記載されているが,その機能を実現するために必要となる２次的な機能について明確になっていないことが多く，精度の高い見積を行うことは難しい場合が多い．ここで見積る開発費，開発工数，開発期間は企画段階のものよりも精度は高いものの，概算というべきものになる．

(c)業務要件設計終了以降での見積

購入者，供給者間の契約内容の見直しや，プロジェクトの開発計画をより詳細化すること等を目的に，図 3.1 に示すように，業務要件設計やソフトウェア方式設計完了時に，開発するソフトウェアで実現する機能，性能，信頼性等の機能仕様が明確になった時点で行う見積である.高い精度での見積が可能な場合が多い．

なお，今後，第 3 章では，エンタープライズ系ソフトウェアを単にソフトウェアと呼ぶ．また，見積とはソフトウェアの規模を見積ることを指す．

3.2.2 FP 法の概要と課題

FP 法には，IBM 法[44]，IFPUG 法[19]等、数十種類の計測方法があるが[1],[15],[37],[42],[50]，現在では IFPUG 法が主流となっており，本論文では IFPUG 法を用いる．

(1)IFPUG 法の概要

IFPUG 法には，開発するソフトウェアの規模を表すアプリケーション FP，新規にソフトウェアを開発するプロジェクトの規模を知るために使用する新規開発プロジェクト FP,既存ソフトウェアを機能改良するプロジェクトの規模を知るために使用する機能改良プロジェクト FP の 3 種類がある．

IFPUG 法では，ファンクションポイントを次の手順で計測する[19],[39]．なお，説明の中にでてくるアプリケーションとはエンタープライズ系ソフトウェアと同じ意味である．

Step1：算出種類の選択

アプリケーション FP，新規開発プロジェクト FP，機能改良機能改良プロジェクト FP の 3 種類の中から適切な種類を選択する．

Step2：アプリケーション境界と計測範囲の設定

FP を計測する範囲を明確にするために，計測対象のアプリケーション境界（ソフトウェアの境界）を設定する．アプリケーション境界を今後，計測境界と呼ぶ．

Step3：データファンクションの計測

データファンクションとは，計測対象となるソフトウェアから参照，若しくは更新を行う論理的な意味でのデータのまとまりのことである．データファンクションは，計測対象のソフトウェアから更新される内部論理ファイル(ILF: Internal Logical File)と，計測対象のソフトウェアから参照されるのみの外部インタフェースファイル(EIF: External Interface File)の 2 種類のファンクションタイプに分類できる．

最初に，計測対象のソフトウェアからデータファンクションを抽出し，ファンクションタイプを決定する．次に，それぞれのデータファンクションの複雑度を，データ項目数，レコード種類数の 2 つのパラメータによって，低，中，高の 3 段階に分類する．データ項目とはデータを定義する最小の単位であり，データ項目数は 1 つのデータファンクションの中に存在するデータ項目の個数である．また， 1 つのデータファンクションの中に，異なる意味合いをもつデータの纏まりが混在している場合，その個数をレコード種類数という．混在していない場合は，レコード種類数は 1 となる．

Step4：トランザクションファンクションの計測

ソフトウェアに対するデータの入出力を伴う処理をトランザクションファンクションという．トランザクションファンクションは，計測境界外からのデータ入力によってデータファンクションの更新を行う外部入力(EI: External Input)，計測境界外へ派生データ(計算や条件判断など何らかの加工を必要とするデータ項目)の出力を行う外部出力(EO: External Output)，計測境界外へデータをそのまま出力する外部照会(EQ: External Inquiry)の 3 種類のファンクションタイプに分類できる．

最初に，計測対象のソフトウェアからトランザクションファンクションを抽出し，ファンクションタイプを決定する．次に，それぞれのトランザクションファンクション の複雑度を，関連ファイル数，データ項目数の 2 つのパラメータによって，低，中，高の 3 段階に分類する．関連ファイル数とは，対象となるトランザクションファンクションで参照，若しくは更新するデータファンク

ションの個数である．データ項目数は，実際に参照，若しくは更新するデータ項目の個数である．

Step5：未調整ファンクションポイントの算出

表 3.1 未調整 FP 算出表

タイプ	低	中	高	合計
ILF	a1×7	a2×10	a3×15	ILF の合計
EIF	b1×5	b2×7	b3×10	EIF の合計
EI	c1×3	c2×4	c3×6	EI の合計
EO	d1×4	d2×5	d3×7	EO の合計
EQ	e1×3	e2×4	e3×6	EQ の合計
				総計 (未調整 FP)

Step3, Step4 の結果を基に,表 3.1 を用いて未調整ファンクションポイント(未調整 FP）を計算する．

表 3.1 における a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3, e1, e2, e3 は，次の値である．

・a1, a2, a3：計測対象ソフトウェアに含まれる複雑度が低，中，高の ILF の数
・b1, b2, b3：計測対象ソフトウェアに含まれる複雑度が低，中，高の EIF の数
・c1, c2, c3：計測対象ソフトウェアに含まれる複雑度が低，中，高の EI の数
・d1, d2, d3：計測対象ソフトウェアに含まれる複雑度が低，中，高の EO の数
・e1, e2, e3：計測対象ソフトウェアに含まれる複雑度が低，中，高の EQ の数

Step6：調整係数の算出

未調整 FP には性能，信頼性，ユーザインタフェースらのシステム特性を反映していないので，これらを反映した FP を求めるために，表 3.2 に示す 14 項目のシステム特性の影響度を，0～5 の 6 段階で評価して，調整係数を次の式で算出する．　　調整係数＝0.01×影響度の合計＋0.65

Step7：最終ファンクションポイントの算出

未調整 FP と調整係数から最終 FP を算出するが，算出種類によって算出式が異なる.本研究の対象であるアプリケーション FP の場合は次の式で算出する．

最終アプリケーション FP＝未調整 FP×調整係数

なお，今後はアプリケーション FP の未調整 FP のみを対象にし，未調整 FP を

単に FP と呼ぶ.

表 3.2　システム特性の 14 項目

1	データ通信機能	6	オンラインデータ入力	11	インストレーションの容易さ
2	分散データ処理	7	エンドユーザの効率	12	運用の容易さ
3	性能条件	8	オンライン更新	13	複数サイト
4	高負荷構成	9	複雑な処理	14	変更の容易さ
5	トランザクション処理	10	再利用性		

(2)IFPUG 法の課題

どの時点の見積も重要であるが，特に開発の前段階での見積が重要であると考えられる．この時点での見積が誤って過小になった場合，実際にかかった開発費，開発工数，開発期間は計画したものよりも大きくなる．その結果，例えば，購入者は新しいソフトウェアの稼動開始時期が遅れて事業の好機を逃すことになり,供給者は購入者から支払われる金額を超える開発費がかかり,所謂，赤字になる．即ち，見積を誤った場合の影響が大きい．また，筆者らが，開発費が当初の計画よりも超過したプロジェクトの原因を分析したところ，見積不良によるものが非常に大きな割合を占めていることが判明した．したがって，開発の前段階で精度の高い見積を行うことが重要となる．この時点では，RFP に記載されている機能要件を基に見積を行う必要がある．しかし，記載されている機能に漏れがある場合や，業務要件設計やソフトウェア方式設計が完了していないために詳細部分が固まっていない場合が多い．したがって，IFPUG 法に習熟していない人が計測しようとした場合，次のような課題があると考えられる．

(a)Step3 でのデータファンクションの複雑度を分類するときのパラメータであるレコード種類数，データ項目数を正確に把握することが難しい．

(b)Step4 でのトランザクションファンクションの複雑度を分類するときのパラメータである関連ファイル数，データ項目数を正確に把握することが難しい．

(c)Step4 での EI, EO, EQ の 3 種類のファンクションタイプは，データの入出力を伴う処理を，最も小さな単位に細分化したものであり，抽象的で，具体的な処理や操作との対応づけが難しい．その結果，トランザクションファンクションの抽出が難しい．

このような IFPUG 法の課題を改善するものとして，NESMA 法[12]，電中研法[45]等いくつか考案されている．

NESMA 法には，FP 概算法と FP 試算法の 2 種類の見積方法がある．

FP 概算法は，データファンクションやトランザクションファンクションの抽出及びファンクションタイプの決定の仕方は，IFPUG 法と同じであるが，複雑度をデータ項目数，レコード種類数，関連ファイル数で分類するのでなく，それぞれ予め設定されているデフォルト値を当てはめる．したがって，IFPUG 法での複雑度の分類が難しいという課題は解決されている．しかし，実際のエンタープライズ系ソフトウェアでは，3 種類のトランザクションファンクションは，低，中，高，いずれの複雑度もとりえるが，デフォルト値が一意に当てはめられるために，精度が粗くなると考えられる．また，IFPUG 法と同様に，トランザクションファンクションの抽出が難しいと考えられる．

FP 試算法は，データファンクションの計測は FP 概算法と同じであるが，トランザクションファンクションの抽出を行わず，データファンクションの ILF, EIF の個数から，FP=35×(ILF の数)+15×(EIF の数) により FP を算出する．したがって，FP 概算法よりも更に精度が粗いと考えられる．

電中研法は，表 3.3, 表 3.4 で示すように，機能の要素として画面，帳票，ファイル，電文を，入力，出力，入出力に分類し，それぞれの難易度を単純，普通，複雑の 3 段階で評価し FP を算出する．

表 3.3 データ型と FP 基準値

データ型		FP 基準値		
		単純	普通	複雑
ファイルまたは電文	入力	3	5	8
	出力	3	5	8
	入出力	3	6	13
画面	入力	3	6	13
	出力	3	5	8
	入出力	6	11	19
	メニュー	2	2	3
帳票	―	3	6	10

電中研法は，IFPUG 法のように，データファンクションについてはデータ項目数や関連ファイル数を評価せず，トランザクションファンクションについてはデータの入出力で分類するのでなく，画面，帳票，電文を対象とするので，

わかりやすく，早期の見積に適しているが，IFPUG 法とは互換性がないのが問題と考えられる．

表 3.4 難易度判定表

		処理の複雑度		
		易しい	普通	難しい
項目数	少ない	単純	単純	普通
または	普通	単純	普通	複雑
行種類数	多い	普通	複雑	複雑

本論文では，開発の前段階でも高い精度で見積を行えるように IFPUG 法を改善した，「要素見積法」を提案する．要素見積法は FP 概算法と類似の手法である．

3.3 要素見積法

要素見積法では，3.2 節であげた IFPUG 法の課題の改善のために，データファンクションの複雑度の分類の簡易化と，トランザクションファンクションの抽出及び複雑度の分類の簡易化を図っている．

(1)データファンクションの計測

IFPUG 法と同様に，データファンクションを抽出し，ILF, EIF のいずれかにファンクションタイプを決定する．ただし，複雑度をレコード種類数，データ項目数の 2 つのパラメータによって分類するのでなく，FP のデフォルト値（ILF:8, EIF:5）を FP 単価として用いる．

(2)トランザクションファンクションの計測

要素見積法では，トランザクションファンクションのファンクションタイプを EI, EO, EQ の 3 種類に分類するのでなく，表 3.5 に示す，要素機能と呼ぶ 16 種類のファンクションタイプに分類する．

要素機能は，IFPUG 法ではトランザクションファンクション EI, EO, EQ の抽出が難しいので，これらを想起しやすくするための処理を用意しようという着想から生まれた．例えば，EI として抽出される処理には，「新規登録」，「既存データ変更」，「既存データ削除」があり，これらを要素機能として設定した．IFPUG 法の EI, EO, EQ はデータの入出力を伴う処理を最小の単位に細分化したものであるが，要素見積法での 16 種類の要素機能は，画面上の 1 個の GUI ボ

タンや 1 回のファンクションキーを押すことで実行される具体的な処理であり，ユーザ視点での入出力の最小単位と考えることができる．

表 3.5 要素機能一覧

	要素機能名	機　能　概　要	FP 単価
更新系	新規登録	登録データ入力後，マスタファイルへの新規データ登録．例えば SQL の insert 文．	5
	既存データ変更	変更データ入力後，マスタファイルのデータ変更．例えば SQL の update 文．	5
	既存データ削除	削除データの指示により，マスタファイルのデータ削除．	4
	マスタメンテナンス	マスタファイルの内容をメンテナンスする機能．	12
	その他更新	更新データ入力後，追加，変更，削除が混在する更新．	6
画面出力系	問合せ応答	画面上からの条件入力により，データを表示．	5
	一覧照会	画面上からの条件入力により，該当データの一覧を表示．	5
	明細照会	画面上で一覧から選択により，該当データの詳細内容を表示．	5
	計算結果表示	画面上からデータ入力後，計算結果を表示．	6
	更新のための照会	変更や削除のために，前もって現在値を照会．	4
	選択肢一覧	ListBox や PopUp らで商品コード，商品名の選択肢を表示．	3
	その他照会	上記にあてはまらない照会機能	5
その他出力系	帳票出力	印刷帳票の出力，PDF ファイルの作成，EUR 帳票出力のためのデータの引渡し．	6
	CSV 出力	電子ファイルの出力機能．	5
	その他データ出力	帳票や電子ファイル以外のデータ出力	5
	他システムへの出力	他システム送信用のファイル作成，他システム送信用の電文の作成，API 呼び出しによる他システムへのデータ引渡し．	5

要素見積法でのトランザクションファンクションの計測では,最初に要素機能を抽出する．これは，IFPUG 法におけるトランザクションファンクションの抽出と，ファンクションタイプの決定に該当する．要素機能の複雑度は，IFPUG 法のように，関連ファイル数やデータ項目数で分類するのでなく，表 3.5 の「FP 単価」の列に示す，各要素機能の 1 個あたりの FP のデフォルト値を当てはめる．したがって，IFPUG 法と同様に，FP を算出することができる．

(3)要素見積法での FP 単価の設定と評価

データファンクション ILF, EIF の FP 単価，要素機能の FP 単価は，次のよう

な考え方で設定した.

著者らが所属する組織では,1996 年～1998 年にかけて実施されたプロジェクトの IFPUG 法による実績から，データファンクション，トランザクションファンクションの FP を，複雑度を評価せずに，ILF:8, EIF:5, EI:5, EO:6, EQ:4 と FP 単価を設定すると,実用上,有用であることが経験的に得られていた．そこで，要素見積法における FP 単価は，これらの値を基に設定した.

(a)データファンクションの FP 単価

ILF, EIF の FP 単価は上記の値をそのままを採用し，ILF:8, EIF:5 とした．また，これらの FP 単価が妥当であるかを，金融分野のソフトウェアから 1 つ、製造分野のソフトウェアから 2 つ、公共分野のソフトウェアから 3 つの計 6 つのソフトウェアの IFPUG 法と要素見積法による実測データを用いて評価した. IFPUG 法による実測結果では, ILF, EIF の FP の平均値は, それぞれ 7.3, 5.0 であり，要素見積法の FP 単価との誤差は 10%未満であり，実用的に問題がないため，要素見積法で設定したデータファンクションの FP 単価は妥当であると判断した.

(b)要素機能の FP 単価

各要素機能の FP 単価は，次のような考え方で設定した.

要素機能はデータの入出力を伴う処理であるので, その中には IFPUG 法でのトランザクションファンクション EI, EO, EQ が含まれる．そこで，各要素機能でのトランザクションファンクションの出現頻度と FP を表 3.6 のように想定して，要素機能の FP 単価を設定した. また，ここでの EI, EO, EQ の FP は，先に説明した経験的に得られた EI, EO, EQ の FP 単価，即ち EI:5, EO:6, EQ:4 を基に設定した.

例えば「新規登録」はデータファンクションの更新があり，EI のみが必ず 1 回出現するので，FP は EI の FP 単価の 5 を，そのまま採用した．同様に「既存データ削除」は，EI のみが必ず 1 回出現するが，削除処理ではデータ項目数が少なく複雑度が低くなる傾向にあるので FP には 4 を採用した.「問合せ応答」は EO のみが 1 回出現するものと，EQ のみが 1 回出現するものが，ほぼ同等であるので, EO, EQ の出現頻度を 0.5 ずつに設定した. また, FP は EO, EQ の FP 単価をそのまま採用した．マスタメンテナンスは登録，変更，削除の EI が計 3 回と EQ が 1 回出現するが，複雑度が低いので FP にはそれぞれ 3

を採用した.

表 3.6 要素機能毎の FP 単価算出表

		新規登録	既存データ変更	既存データ削除	マスタメンテナンス	その他更新	問合せ応答	一覧照会	明細照会	計算結果表示	更新のための照会	選択肢一覧	その他照会	帳票出力	CSV出力	その他データ出力	他システムへの出力
EI	出現頻度	1.0	1.0	1.0	3.0	1.0											
	FP	5	5	4	3	6											
EO	出現頻度						0.5	0.5	0.5	1.0			0.5	1.0	0.5	0.5	0.5
	FP						6	6	6	6			6	6	6	6	6
EQ	出現頻度				1.0		0.5	0.5	0.5		1.0	1.0	0.5		0.5	0.5	0.5
	FP				3		4	4	4		4	3	4		4	4	4
要素機能の FP 単価		5	5	4	12	6	5	5	5	6	4	3	5	6	5	5	5

このように，IFPUG 法による実測の経験から表 3.6 を作成し要素機能の FP 単価を設定したが，これらの FP 単価の妥当性について，データファンクションの FP 単価と同様に 6 つのソフトウェアについて，IFPUG 法と要素見積法による実測データを比較することで評価した．その結果を表 3.7 に示す.

FP 単価の評価では，表 3.7 における FP 単価－FP 平均値が FP 平均値の-10%～+25%の範囲に納まっているかを妥当であるかの基準とした[12]．したがって，「その他更新」，「更新のための照会」，「その他照会」，「CSV 出力」を除いて，FP 単価は妥当なものであると判断した．この範囲に納まっていない要素機能については，出現頻度を加味して評価した．例えば「その他更新」は+57.9%と大きく外れているが，一方，要素機能の出現頻度の平均値が 6.7%に対して「その他更新」の出現頻度は 1.8%と少なく，実質的な影響は少ない．同様に「更新のための照会」，「その他照会」，「CSV 出力」も実質的な影響は少ないと評価できる． これらの結果より，我々が対象としているドメインにおいては，設定した FP 単価の値は妥当なものであると判断した.

表 3.7　6 つのエンタープライズ系ソフトウェアの実測結果

要素機能	個数	出現頻度(%)	FP 合計	FP 比率(%)	FP 平均値	FP 単価	(FP 単価-FP 平均値)／FP 平均値(%)
新規登録	580	11.8	2526	12.0	4.4	5	13.6
既存データ変更	712	14.5	2980	14.2	4.2	5	19.0
既存データ削除	439	9.0	1671	8.0	3.8	4	5.3
その他更新	87	1.8	334	1.6	3.8	6	57.9
問合せ応答	537	11.0	2362	11.2	4.4	5	13.6
一覧照会	565	11.5	2564	12.4	4.6	5	8.7
明細照会	131	2.7	692	3.3	5.3	5	-5.7
計算結果表示	12	0.2	61	0.3	5.1	6	17.6
更新のための照会	94	1.9	455	2.2	4.8	4	-16.7
選択肢一覧	650	13.3	2045	9.7	3.1	3	-3.2
その他照会	17	0.3	59	0.3	3.5	5	42.9
帳票出力	586	12.0	3170	15.1	5.4	6	11.1
CSV 出力	133	2.7	433	2.1	3.3	5	51.5
その他データ出力	294	6.0	1285	6.1	4.4	5	13.6
他システムへの出力	61	1.2	329	1.6	5.4	5	-7.4
合計	4898	100.0	20996	100.0	65.5	74	221.8

なお，これまでの説明から，要素見積法で測定した FP と IFPUG 法で測定した FP は，精度を別にすれば，基本的には同じものとなることが理解できる．

3.4　要素見積法のシステム化(AP-Estimate)

見積の容易化，見積結果の蓄積らを目的に，見積支援システム AP-Estimate を開発した．本章では AP-Estimate の構成，機能について説明する．

3.4.1　AP-Estimate の構成

図 3.2 に AP-Estimate の構成を示す．

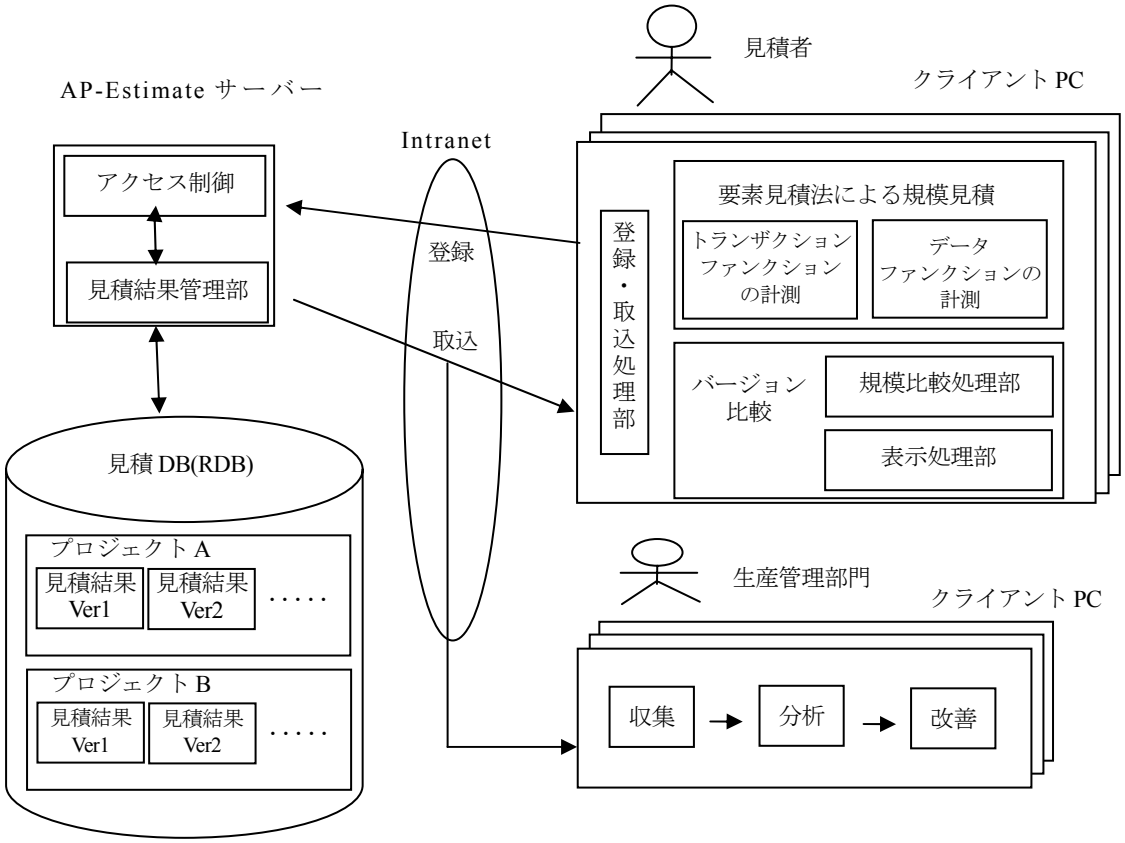

図 3.2 AP-Estimate の構成

AP-Estimate は図 3.2 で示すように，1 台のサーバーと複数のクライアント PC から構成され，イントラネットを介して通信を行う．サーバーには見積 DB が接続されている．見積を行うプロジェクト管理者や開発者は，クライアント PC から必要な情報を入力して見積結果（FP）を得る．また，この結果は見積 DB に格納される．見積技法の開発を担当している生産管理部門は，見積 DB に格納されている多くの見積結果を分析し，例えば要素機能の FP 単価へのフィードバックらの見積精度の改善を実施する．

3.4.2 AP-Estimate の機能

(1)要素見積法のサポート

利用者は，図 3.3 で示すように，クライアント PC から各要素機能の個数を入力することによりトランザクションファンクションの FP を算出できる．同様に図 3.4 で示すように，ILF, EIF の個数を入力することによりデータファンクションの FP を算出できる．

図 3.3 トランザクションファンクションの FP 算出のための入力例

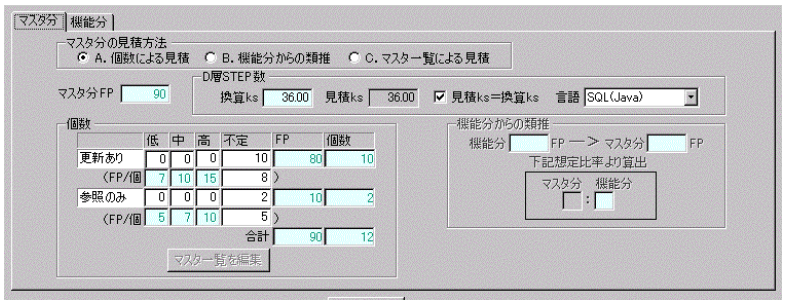

図 3.4 データファンクションの FP 算出のための入力例

(2)見積結果の蓄積

見積結果を見積 DB に格納する．見積はプロジェクト毎に実施するが，企画段階，開発の前段階，業務要件設計終了以降等、複数回，実施することが多いと考えられるので，プロジェクト名，格納した年月日及び時刻を見積結果に付加し，区別できるようにした．

見積実施時期で区別された各見積結果をバージョンと呼ぶ．なお，業務要件設計終了以降は IFPUG 法によって見積を行うことが多いと考え，AP-Estimate では IFPUG 法による見積結果も蓄積できるようにしている．

(3)バージョン比較

バージョン間を比較し，差異を分析することにより，より精度の高い見積を実施できる．

(4)アクセス制御

見積結果にパスワードを付加して登録することにより，特定の人以外の参照

を禁止することができる.

3.5 適用事例

一般にRFPに記載されている機能要件ではソフトウェアの機能仕様は明確でなく，IFPUG 法による見積は難しいと考えられている．3.5 節では，RFP に記載されている機能要件から要素見積法で規模の見積を行えることを，在庫管理ソフトウェアでの実例を用いて説明する.

3.5.1 在庫管理ソフトウェアの機能要件

データに関する機能要件，業務に関する機能要件を図 3.5 に示す．ただし，図 3.5 は，業務仕様に関しては，商品管理，入庫管理，選択肢一覧についてのみ，機能要件を示しており，他は省略してある.

3.5.2 FP の計測

(1)データファンクションの計測

図 3.5 に記載されているデータに関する機能要件から，ILF，EIF の個数は，それぞれ 10, 2 であることがわかるので，データファンクションの FP は，10×8＋2×5＝90 と算出できる.ここで 8, 5 は,それぞれ ILF, EIF の FP 単価である.

(2)トランザクションファンクションの計測

共通機能である選択肢一覧と商品管理の FP を算出する．算出のやり方は，図 3.5 に記載されている機能要件に対して，見積者が更に想定する仕様を追加し，要素機能を抽出して，個数を数えるものである.

(a)選択肢一覧

・機能要件「2.商品はいくつかのカテゴリー，サブコテゴリーに分類され，入出庫情報の登録，更新時にカテゴリーやサブカテゴリーの一覧から選択する.」には，カテゴリーから選択，サブカテゴリーから選択の 2 種類の処理があり，それぞれが要素機能「選択肢一覧」に該当するので,「選択肢一覧」が 2 個存在する.

・機能要件「3.入出庫情報の登録，更新時に，商品を一覧から選択する.」には，要素機能「選択肢一覧」が 1 個存在する.

・機能要件「4.入出庫情報の登録，更新時に，在庫のある商品一覧から商品を

選択する.」には，要素機能「選択肢一覧」が 1 個存在する.

・したがって，選択肢一覧には，要素機能「選択肢一覧」が 4 個存在する．表 3 より，選択肢一覧の FP 単価は 3 ゆえ，中機能選択肢一覧の FP は，4×3＝12 となる.

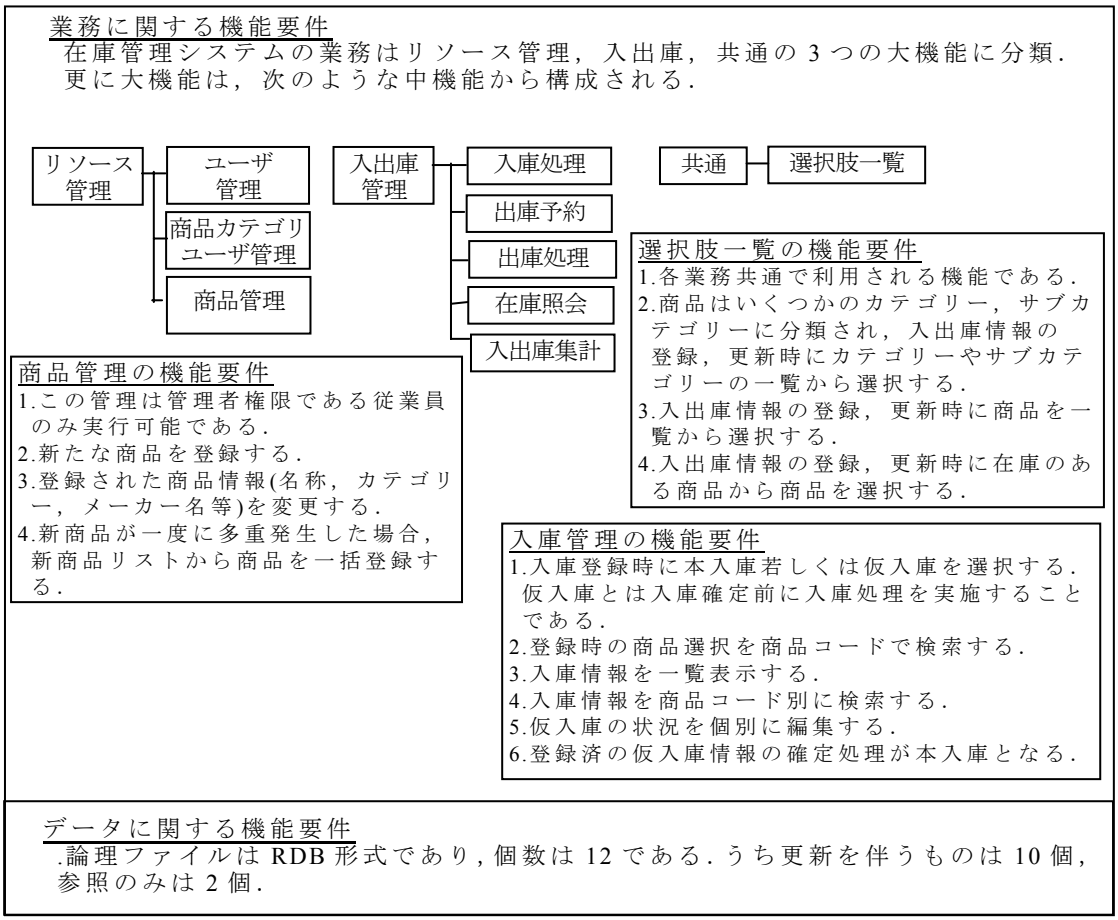

図 3.5 在庫管理ソフトウェアの機能要件

(b)商品管理

・機能要件の「2.新たな商品を登録する.」には，要素機能「新規登録」が 1 個存在する.

・機能要件の「3.登録された商品情報（名称，カテゴリー，メーカー名ら）を変更，削除する.」には，商品を選択し，その商品の登録情報を照会し，変更若しくは削除を行う処理が想定される．商品の選択のうち，カテゴリー，サブカテゴリーからの選択は共通機能の選択肢一覧に該当する.したがって，要素機能「一覧照会」,「明細照会」,「既存データ変更」,「既存データ削除」が各 1 個ずつ存在する.

・機能要件「4.新商品が一度に多重発生した場合，新商品リストから商品を一

括登録する.」は，要素機能「新規登録」が 1 個存在する.

・したがって，要素機能「新規登録」が 2 個，「既存データ変更」が 1 個，「既存データ削除」が 1 個，「一覧照会」が 1 個，「明細照会」が 1 個であり，FP は 2×5＋1×5＋1×4＋1×5＋1×5＝29 となる.

他の中機能についても，同様の方法で FP を算出できる.

3.6 評価

(1)要素見積法の精度

要素見積法の精度の評価では，要素機能の FP 単価の精度評価と，要素機能の抽出精度の 2 つの面から行う必要がある.

(a)FP 単価の精度

要素見積法では,各要素機能の FP 単価を表 3.5 の FP 単価の列に示すように設定している．したがって，この FP 単価の設定値が要素見積法の精度に影響を与えると考えられる．この精度分析も，3.3 節で要素機能の FP 単価の評価と同時に行った.

図 3.6 は，6 つのエンタープライズ系ソフトウェアについて，IFPUG 法で計測した FP を 100%として，要素見積法で算出した FP を評価したものである.図 3.6 では, FP を更にデータファンクションの FP(DF FP)とトランザクションファンクションの FP(TF FP)に分解して表示している.

図 3.6 に示すように，要素見積法は，IFPUG 法に対し，-4%～+11%　の範囲に納まっていることを確認できた.

また，要素見積法が IFPUG 法より大きくなる傾向にあることが図 3.6 からわかる. 特にデータファンクションは 6 つのソフトウェア全てで大きく，トランザクションファンクションも 5 つのソフトウェアで大きい. 6 つのソフトウェアを IFPUG 法で計測すると，ILF の平均値は 7.3 であり，EI, EO, EQ の平均値は，それぞれ 4.3, 5.9, 4.0 である．一方，3.3(2)項で説明したように，要素見積法での FP 単価設定では，ILF:8, EI:5, EO:6, EQ:4 を用いており，要素見積法は IFPUG 法よりも，FP 単価が大きい．これが 6 つのソフトウェアにおいて，要素見積法が IFPUG 法より大きくなる理由である.

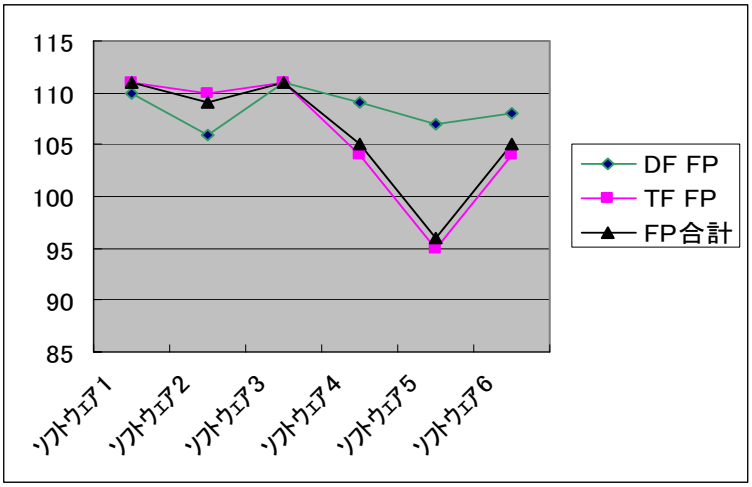

図 3.6 要素見積法の精度分析

(b)要素機能の抽出精度

要素見積法では，開発の前段階において，RFP に記載されている機能要件から要素機能を精度高く抽出できることを，出張旅費精算を行うソフトウェアを例にとり評価する．

表 3.8 旅費精算ソフトウェアの RFP

機　　能
1)事前に登録しておいた出張案件から，精算したい案件を選択する．
2)旅費に関する事項を入力しシステムに登録する．
3)登録した旅費精算の内容を後で修正できる．
4)領収書の提出などのために，旅費精算書という帳票を出力する．

RFP では，出張旅費精算ソフトウェアの機能要件について，表 3.8 の程度しか示されていないことは，よくあることである．IFPUG 法や NESMA FP 概算法では，この情報からトランザクションファンクションを抽出することになる．

表 3.9 IFPUG 法または NESMA 法による未習熟者の見積結果

機　　能	見積結果
1)事前に登録しておいた出張案件から精算したい案件を選択する．	EQ×1
2)旅費に関する事項を入力しシステムに登録する．	EI×1
3)登録した旅費精算の内容を後で修正できる．	EI×1
4)領収書の提出などのために，旅費精算書という帳票 を出力する．	EO×1
	FP 合計:17

FP 合計の算出では EI, EO, EQ の複雑度を”中”と仮定している．

手法に習熟していない人は，表 3.9 のように見積る可能性が高く，これでは過小見積となってしまう。一方，見積手法に習熟している人は，機能の説明とトランザクションファンクションの関係を把握して，表 3.10 のような適切な見積を行うが，ここまで習熟するには一定の経験を必要とする．

表 3.10 IFPUG 法または NESMA 法による習熟者の見積結果

習熟者の考え方	見積結果
1)精算したい案件を選択するには，まず，条件入力→候補一覧というステップがあり，続いて 1 件選択→ 精算案件内容表示となる．即ちここには２つの照会が存在し，更に前者は計算を伴う可能性が高いので EO と判断できる．	EO×1 EQ×1
2)文面どおり，登録のトランザクションファンクションがあると判断する．	EI×1
3)修正するには登録した内容を表示する必要がある． その表示は 1)のケースと同様に 2 段階の照会となる． また，修正には削除機能も含まれると解釈できる．	EO×1 EO×1 EI×2
4)帳票の出力は通常計算を伴うので EO と判断できる．	EO×1
	FP 合計:36

FP 合計の算出では EI, EO, EQ の複雑度を”中”と仮定している．

IFPUG 法や NESMA 法で，このような習熟した考え方にたどりつけない要因として以下の 2 点があると考えられる．

・「2 段階の照会」,「修正の陰に削除あり」等，いわゆる機能説明の行間にあたるものを，EI，EO，EQ のファンクションタイプの定義だけから想起することは容易でない．

・開発の前段階での見積では，EOかEQかの判別は難しい．何故ならEQではなくEOとなる要件は「処理中に計算が含まれる」か「データファンクション更新を伴う照会である」のいずれかであるが，いずれも，その時点では明確でないことが多い．

要素見積法での見積結果は表 3.11 のようになる．

表 3.11 で示すように，要素機能は具体的な機能を表すので，その機能に関連する要素機能が想起しやすく,要素機能の抽出漏れを防止するナビゲーションとしての役割を果たしていることがわかる．このように，IFPUG 法や NESMA 法等に比べて,要素見積法は見積の初心者であってもトランザクションファン

クションの抽出を容易に行うことができる.

表 3.11 要素見積法による見積結果

要素見積法での考え方	見積結果
1)「案件の選択」とあるので案件を表示する“明細照会”が洗い出せるが“明細照会”があるなら“一覧照会”も存在するのではないか,ということが想起される.	一覧照会 明細照会
2)旅費精算の登録は“新規登録”であるが“新規登録”があると“既存データ変更”や“既存データ削除”を伴うことが多い。変更や削除は 3)で示されているのでそちらで計上する.	新規登録
3)修正が“既存データ変更”で,変更があれば“既存データ削除”も存在すると想起される。また“更新のための照会”という要素機能種別により,変更・削除には登録内容の照会を伴うことが想起され,それが 1)のケースと同様に 2 段階の照会となるかどうかという連鎖思考に結びつく.	既存データ変更 既存データ削除 一覧照会 明細照会
4)帳票の出力は EQ である可能性もある。早期見積時点では EO か EQ かは判断がつかないことが多いが,要素見積法では EO か EQ かは判断せず“帳票出力”として見積る.	帳票出力
	FP 合計:40

(2)その他の評価

AP-Estimate により規模の見積を実施した開発者に,AP-Estimate の効果についてアンケートをとり,次のような評価を得た.

(a)各要素機能の個数の分布を評価することで,特定の要素機能の個数が多すぎる,少なすぎるということが判断でき,見積誤りを防止できる.

(b)IFPUG 法では,トランザクションファンクション,データファンクションの複雑度の評価に時間を費やすが,要素見積法では複雑度の評価が不要である.実際にソフトウェアの FP を,IFPUG 法と要素見積法の 2 つの技法で計測したところ,要素見積法の計測速度は IFPUG 法に対して 3 倍～5 倍速かった.

3.7 結論

本論文では,IFPUG 法による見積が難しいと考えられている開発の前段階でも見積が可能な要素見積法の提案と,それをシステム化した AP-Estimate の開発を行った.3.6 節で示すように,要素見積法は,1)精度が IFPUG 法の-4%～+11%

の範囲に納まっており，2)仕様の明確でない開発の前段階でも要素機能の抽出が容易であり，3)計測速度が IFPUG 法の 3 倍～5 倍速く，開発の前段階での見積技法として，十分なものであることを確認できた．また，見積支援システム AP-Estimate は，膨大な量の見積結果の蓄積ができる．しかも，多くのソフトウェアの見積結果,同一のソフトウェアのライフサイクルの各時点での見積結果,要素見積法や IFPUG 法ら異なる見積技法による見積結果が蓄積される．したがって，これらの膨大なデータを分析することで，より精度が高く，適用の容易な見積技法への改善が期待できる．今後は，見積結果及び実績に関するデータを十分に蓄積し，分析を行い，より精度の高い見積技法に改善していきたい．

第 4 章　WBS に基づくプロジェクト管理システムの実現

4.1　緒言

インターネットや Web の普及拡大に代表されるように情報化社会はますます進んでおり，それに伴い，ソフトウェアの開発においても，大規模化，高機能化，短納期化，低コスト化の要求が急速に高まってきている．それに応じて，エンタープライズ系ソフトウェアの開発プロジェクトを計画通りに達成するために，プロジェクト管理の重要性も急速に高まってきている[14],[20],[21],[23],[35],[48],[49],[58]．プロジェクトは，複数の作業とその結果である複数の成果物から成り立っていると考えられる．したがって，プロジェクト管理とはそれらの作業を効率よく遂行できるようにすることと，作業や成果物を管理することと考えることができる．プロジェクト管理では，各作業や成果物に対して計画を立て，遂行し，その進捗状況をチェックし，問題があれば対策を取るという，いわゆる Plan-Do-Check-Action を繰り返し行うことが必要である．しかし，大規模なエンタープライズ系ソフトウェアの開発プロジェクトにおいては，作業や成果物が膨大な数に渡るほか，多数のプロジェクトメンバが複数の開発拠点に分散しており，プロジェクトの計画や状況，成果物等の情報共有が難しい．また，作業や成果物作成にあたって設計標準等の利用する資料も膨大であり，適したものを捜す負荷も大きい．このため，プロジェクト管理を人手で実施するのでは非常に難しく，支援するシステムが必要とされている．

そこで本章では，WBS(Work Breakdown Structure)モデル[3],[16],[22],[47]に基づいた作業や成果物の管理方法やプロジェクト管理支援方法を提案する．また，実際に開発した，WBS モデルに基づくプロジェクト管理システム「プロナビ」について述べる．

プロジェクト管理のシステム化については，様々な研究が行われてきている．また，Microsoft Project [55]，ProcessDirector[46]，PMOffice Enterprise[25]等，システム管理を行うツールも数多く存在する．しかし，これらの現実の大規模プロジェクトへの適用事例の報告は少なく，適用可能性はあまり議論されていない．

我々は，大規模プロジェクトの管理には，工程，作業の階層化，そして成果物，参考資料等の情報を関連付けが重要であると考え WBS モデルを用いて，プロナビを開発した．また，開発現場への適用を進めてきた．

プロナビでは，それらの情報を WBS モデルにより相互に関連付けし，一元管理することで，1)プロジェクト計画時の工程，作業，成果物の明確化，2)プロジェクト進捗状況の把握，3)プロジェクトメンバ間での成果物の共有，4)規則，標準，ワークシート等の組織に蓄積された知識の活用，5)開発プロセス及び作業の標準化を実現しており，プロジェクト管理の効率化に役立っている．

プロナビは現在までに,延べ 2000 を超えるプロジェクトで適用された実績があり，今後も増加する見込みである．

以降，4.2 節で WBS によるエンタープライズ系ソフトウェアの開発プロジェクトのモデル化を，4.3 節でプロナビの実現方式を，4.4 節でプロナビの適用実績と評価及び考察を，4.5 節では関連研究を，4.6 節ではまとめと今後の課題について述べる．

4.2 WBS によるプロジェクトのモデル化

4.2.1 WBS モデル

WBS は，プロジェクトの目標をより具体的に記述するための階層図で，次のように定義されている[16].

(1)WBS は，システム開発していく中で作成されるハードウェア，ソフトウェア，役務，マニュアル，設備を要素とする階層図である．

(2)WBS は開発される成果物を明確化し，また，そのための作業を相互に関連づける．

(3)WBS は必要な階層まで展開できる．

また，[3]では，「WBS はプロジェクト目標を達成するのに必要な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，要素成果物を基にして階層的に要素分解したもの．一段レベルが下がるごとにプロジェクトの作業は更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．ここでワークパッケージとは WBS 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である.」と定義している．ここでは作業を構成要素として階層化を行うこととする．通常，作業とその成果物の対応関係は明白であるため，[3]で述べているこの WBS は，プロジェクトの成果物を要素として階層化した[16]と等価なモデルと考えることができる．本論文では以降，[3]での定義を用いて説明する．また，WBS の各要素をワーク(work)と呼ぶ．

4.2.2 WBS に基づくプロジェクト計画

(1)WBS モデルの考え方

エンタープライズ系ソフトウェアの開発プロジェクトは，設計書等の成果物と，その成果物を作成するための作業から構成されており，WBS モデルを作成することができる．エンタープライズ系ソフトウェアは規模が大きいために，相互の関連が比較的緩いサブシステムに分割して，各サブシステムを開発しやすい規模にすることが多い．このサブシステムを開発するプロジェクトをサブプロジェクトと呼ぶ．

一般にプロジェクトは要求分析のようなプロジェクトの初期時に行う作業からプログラミング＆単体テストのような作業を経てプロジェクトの終了に至る．このように各作業には時間的な順序性がある．そこで，プロジェクトの開始から終了までを，時間的な順序に基づき，フェーズと呼ぶ複数のグループに分割する[32],[35]．これらのフェーズは順を追って終了していくので，フェーズの進捗状況によりプロジェクト若しくはサブプロジェクトの進捗状況を把握することができる．そこでエンタープライズ系ソフトウェアの開発プロジェクトの WBS モデルには，サブプロジェクト，フェーズを取り入れた．

(2)標準プロナビ WBS の作成

エンタープライズ系ソフトウェア開発プロジェクトで用いる WBS モデルとして，次に示す 5 階層から構成されるモデルを提案し，これを標準プロナビ WBS と呼ぶ．

①第 1 階層：プロジェクト

②第 2 階層：サブプロジェクト．

③第 3 階層：フェーズ

④第 4 階層：作業ステップ

⑤第 5 階層：成果物

図 4.1 に標準プロナビ WBS の例を示す．第 1 階層はプロジェクトそのものとする．第 2 階層はプロジェクトを分割したサブプロジェクトとする．

例えば，ある企業の経営管理システムを構築するプロジェクトの場合，第 1 階層は「経営管理システム」が，第 2 階層は経営管理システムのサブシステムである「従業員管理システム」,「経理管理システム」,「商品管理システム」とな

る．

第 3，第 4，第 5 階層は，それぞれ，フェーズ，作業ステップ，成果物とする．これらの作業ステップ，作業項目は，ソフトウェアを中心としたシステム開発および取引のための共通フレーム SLCP-JCF98 における開発プロセスでのアクティビティ，タスクにほぼ対応している[40]．

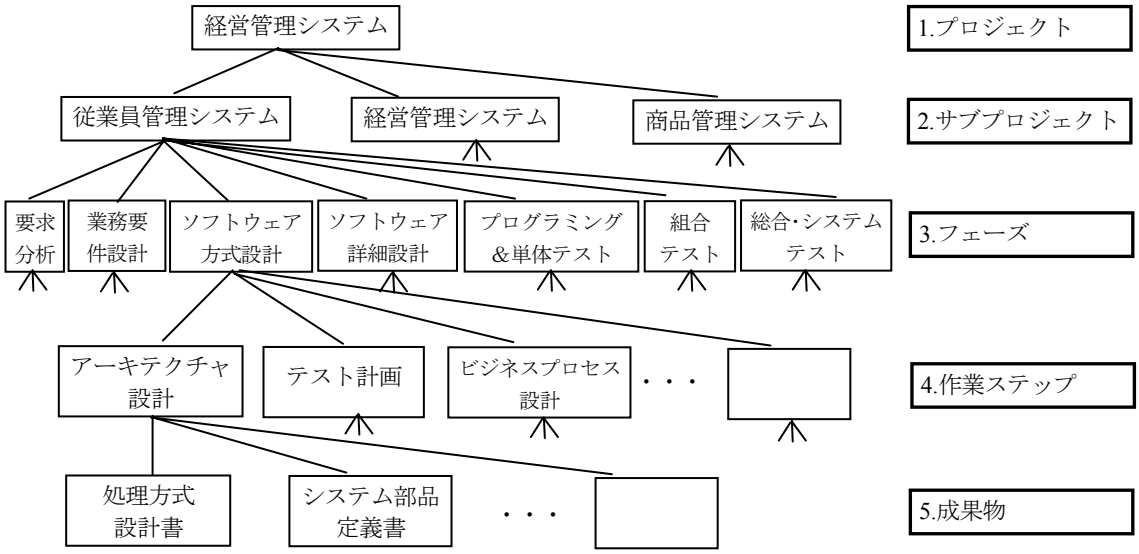

図 4.1 標準プロナビ WBS

標準プロナビ WBS は図 4.1 のように，第 3，4，5 階層のワーク名をあらかじめ定めている．第 3 階層の各フェーズは，要求分析，業務要件設計，ソフトウェア方式設計，ソフトウェア詳細設計，プログラミング＆単体テスト，組合テスト，総合・システムテストの 7 つであり，プロジェクトライフサイクルを分割したものとなる．これらは，時間的にもこの順序で進行する．

第 4 階層の作業ステップは，各フェーズの作業内容を詳細化して，複数の具体的な作業に分割したものである．例えば，ソフトウェア方式設計は，アーキテクチャ設計，テスト計画，ビジネスプロセス設計，データベース層クラス設計，機能層クラス設計，プレゼンテーション層イベント設計，データベース物理設計の 7 つの作業ステップに分割する．

第 5 階層の成果物は，処理方式設計書，システム部品定義書，ビジネスプロセスフロー図，画面遷移図等である．

成果物の実体はファイルで管理される．１つの成果物を複数の担当者で分担して作成すること等を考慮して，1 つの成果物は複数のファイルから構成され

る場合も許す．例えば，処理方式設計書という成果物はオンライン処理方式設計書，バッチ処理方式設計書という 2 つのファイルから構成されるようにできる．このファイルを成果物ファイルと呼ぶが，成果物ファイルはプロナビ WBS 上には表れない．なお，成果物と成果物ファイルを区別する必要がない場合は，単に成果物と呼ぶことにする．

フェーズの間は，その並びのとおりの時間的な順序関係が存在するが，作業ステップ，成果物の間には必ずしも時間的な順序関係は存在しない．

(3)標準プロナビ WBS の適応

プロジェクトの開始時に，プロジェクト管理者が，プロジェクトを管理するのに適した WBS を作成する．通常，[3]のような参考書や組織内に存在する WBS 作成のための標準を基にして作成することが多い．本研究では，標準プロナビ WBS を基にプロジェクトに適した WBS を作成する．しかし，標準プロナビ WBS は，エンタープライズ系ソフトウェア開発の標準的な開発プロセスを定義するものであり，個々のプロジェクトに適用する場合，ワークの過不足が生じることがある．そこで，標準プロナビ WBS に対し，必要なワークの追加，不要なワークの削除，ワーク名の変更等を行い，標準プロナビ WBS を自プロジェクトに適用できるようにする．この WBS を個別プロナビ WBS という．標準プロナビ WBS に対する無制限な変更を許可すると，個別プロナビ WBS と標準プロナビ WBS の構造の間の差異が大きくなり，プロジェクト管理の標準化が難しくなる恐れがある．そこで，特定のワークに対しては省略・変更を制限することで，各プロジェクトの個別プロナビ WBS は成果物名称，作業名称，基本構造がほぼ同一になり，一定の構造を保つようにできる．標準プロナビ WBS と個別プロナビ WBS を特に区別しないとき，単にプロナビ WBS と呼ぶ．プロナビでは，当初，2 種類の標準プロナビ WBS を用意した．

(4)プロナビ WBS に基づくプロジェクト計画

プロジェクトの計画は，プロジェクトの開始日，終了日，プロジェクトメンバ，WBS, 各ワークの開始日，終了日，担当者を決めることで明確になるが，プロナビ WBS を適用することにより，作業や成果物間の，プロジェクトの進行方向での対応関係と，上位レベルから下位レベルへの階層的な方向での対応関係を付けることができる．

プロジェクトのメンバがある作業を遂行するとき，その作業の前提となる作

業での成果物を利用する必要が生じる．また，規則，標準，手順，ワークシート，過去のベストプラクティスのような組織に蓄積された知識（今後，知識情報と呼ぶ）を利用することで，作業を効率よく，かつ均等品質を確保して進めることができる．しかし，成果物は，作成者自身が個人で管理していて，プロジェクト全体での共有や版管理が行われていない場合が多い．また，知識情報は組織内で蓄積されていても，量が膨大な上に，目的別に整理されていない場合が多く，適したものを捜す負荷が大きい．そこで，成果物は電子化してプロジェクトで共有できるデータベースに，知識情報は電子化して組織全体で共有できるデータベースに格納する．更に，プロナビ WBS の各ワークに対し，1)当該ワークの前提となるワーク（前提ワーク）とその実体の格納場所，2)当該ワークを実行するのに利用する知識情報とその実体の格納場所，3)ワークの担当者，期限及びワークの進捗状況（未着手，着手，作成完，審査完，承認完），4)成果物を構成する成果物ファイル及び担当者，更新時刻，バージョン番号，進捗状況の 4 種類の情報を付加する．これにより WBS のワークをキーにして担当者，スケジュール，進捗状況，成果物，知識情報が一元管理されることになる．更にプロナビ WBS の構造がプロジェクトの開発プロセスと一致しているので，プロジェクトのメンバはナビゲーションされながら作業を進めることになる．

図 4.2 は，プロナビ WBS に付加された情報の例である．この例では，「経営管理システム」プロジェクトの基本情報に個別プロナビ WBS とプロジェクトメンバの情報が付加されている．更に，個別プロナビ WBS の各ワークにワーク自身の計画や前提ワーク，参照する共通資料，成果物ファイルに関する情報が付加されている．図 4.2 では，メンバの一人である原田が「業務用語集」を 2004 年 4 月 16 日までに作成予定であり，業務用語集を構成する成果物ファイルの一つである“用語集－製造”の作成を粟根が担当していて，4 月 12 日に更新したことを示している．

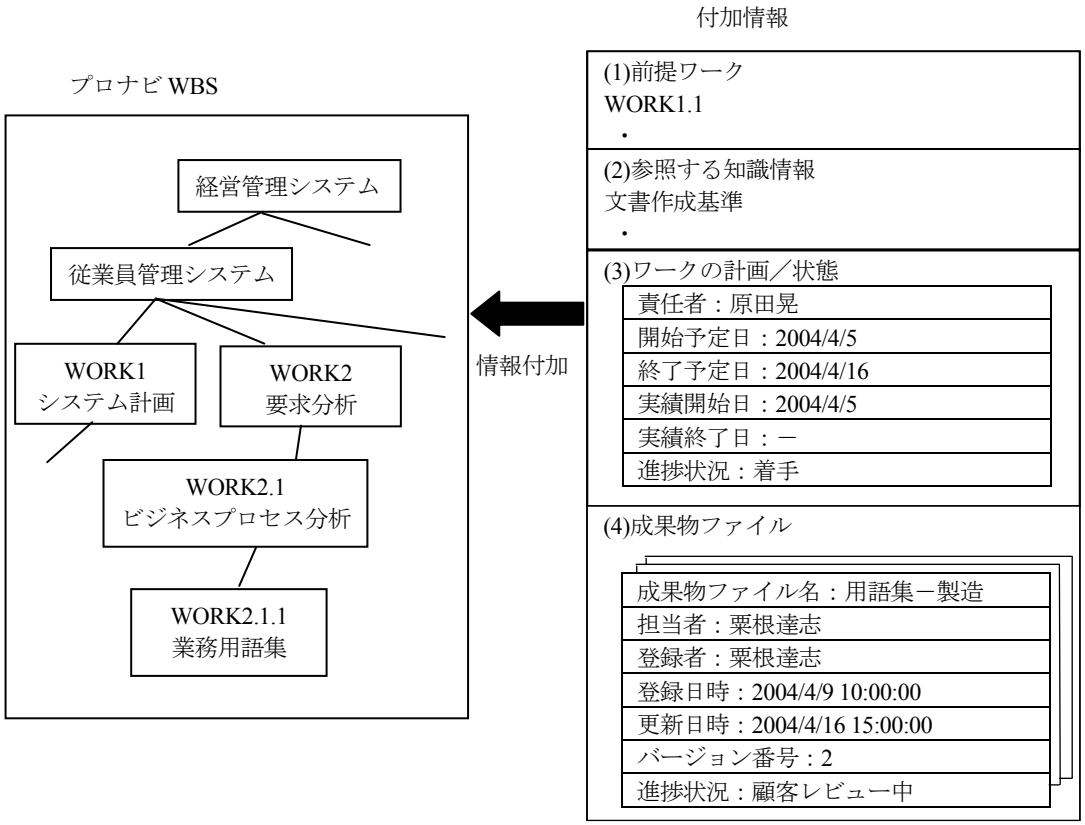

図 4.2 プロナビ WBS への付加情報の例

4.3 WBS に基づくプロジェクト管理システム「プロナビ」

4.3.1 プロナビの構成

大規模なエンタープライズ系ソフトウェアの開発プロジェクトでは，多数のプロジェクトメンバが複数の開発拠点に分散しており，プロジェクトの計画や状況，成果物等の情報共有が難しい．また，参照する資料が膨大であり捜す負荷も大きい．このような課題を解決するものとしてプロジェクト管理システム「プロナビ」を開発した．

プロナビは，プロジェクトの計画，現在の状況，成果物及び知識情報を電子化された文書ファイルとして，サーバで一元管理する．これらのファイルはプロナビ WBS の各ワークと対応がつけられている．プロジェクトメンバのクライアント PC は，サーバとイントラネット若しくはインターネットで接続されており，クライアント PC からワークを指定すると，そのワークの計画，状況，前提となるワークでの成果物一覧，ワークを遂行する時に利用する知識情報一覧が PC 上に表示される．更に成果物や知識情報の内容をクライアント PC 上に

表示させることができる.

プロジェクト管理者は,「4.2.2(4)プロナビ WBS に基づくプロジェクト計画」で説明したように，クライアント PC から，プロナビが用意した複数個の標準プロナビ WBS の中から自プロジェクトに適したものを選び,カスタマイズし，更に各ワークの開始日等の情報を付加してプロジェクト計画を作成できる.

図 4.3 はプロナビ全体の構成を示した図である.

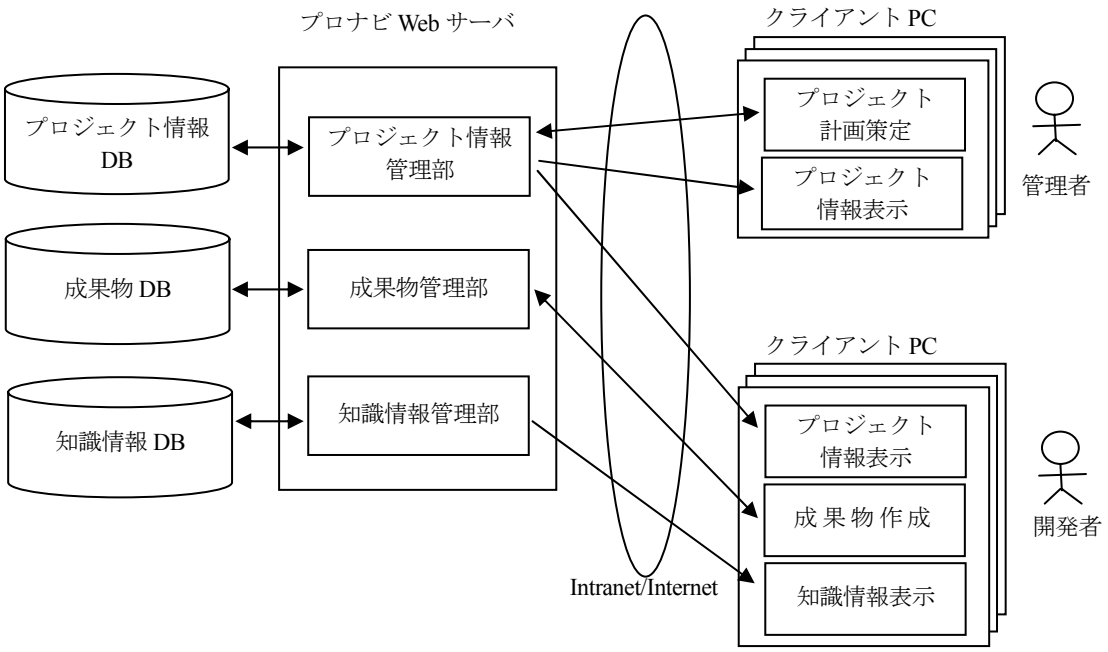

図 4.3 プロナビの構成

プロナビは Web サーバと複数のクライアント PC から構成され，インターネット若しくはイントラネットを介して相互に通信を行う．Web サーバにはプロジェクト情報 DB，成果物 DB，知識情報 DB が接続されている．プロジェクト情報 DB には，図 4.2 で示した情報が格納されている．成果物 DB には成果物の実体である成果物ファイルが格納されている．知識情報 DB には知識情報が格納されている． Web サーバはプロジェクト情報管理部,成果物管理部，知識情報管理部を有している．管理者がクライアント PC で作成したプロジェクト計画は Web サーバに送信され，プロジェクト情報管理部で処理されてプロジェクト情報 DB に格納される．開発者がクライアント PC で作成した成果物ファイルは Web サーバに送信され，成果物管理部によって処理されて成果物 DB に格納される．プロジェクト情報 DB に格納されているプロジェクト情報は，プロジェクトメンバのクライアント PC からの指示により，Web サーバのプロジェ

クト情報管理部によって処理後，クライアント PC に送信されて表示される．同様に，成果物や知識情報の内容はプロジェクトメンバのクライアント PC からの指示により，成果物管理部や知識情報管理部によって処理後，クライアント PC 上に表示される．

4.3.2 プロナビの機能

(1)プロジェクト計画の作成

プロジェクト管理者はプロジェクトの計画を作成するために，自プロジェクトに適した標準プロナビ WBS を選択し，カスタマイズするための情報，ワークの計画，前提ワーク，参照する共通資料，成果物ファイル情報をクライアント PC から入力する．更に，プロジェクト名，期間，プロジェクトメンバを付加したプロジェクト計画は，ネットワーク経由でサーバに送信され，プロジェクト情報 DB に格納される．

(2)プロジェクト情報，成果物，知識情報の表示

プロジェクトメンバのクライアント PC 上にプロジェクト情報，成果物，知識情報を表示する機能として，private view と project view と呼ぶ 2 種類の画面がある．

(a)private view

クライアントPC上を操作しているプロジェクトメンバ自身が担当しているワーク，成果物ファイルの一覧及びその進捗状況を示す画面である．実際の画面例を図 4.4 に示す．

ここでは,あるメンバの参加している複数のプロジェクトまたはサブプロジェクトにおいて，担当者として登録されているワークと，作成，登録した成果物ファイルの一覧が表示されている．図中の「プロジェクトパス」はプロナビ WBS の 1，2 階層を表しており,「ワークパス」は 3 階層以下を表している．

図 4.4 における最初の項目は,「事業統合化プロジェクト」プロジェクトのワーク「非機能要件(システム要件)の定義」の担当であり，“2004/12/10”が期限で,「着手」している状態であることを示している．

一人のメンバが複数のプロジェクトに属している場合は,属しているプロジェクトで担当しているワーク，成果物ファイルの一覧が表示される．

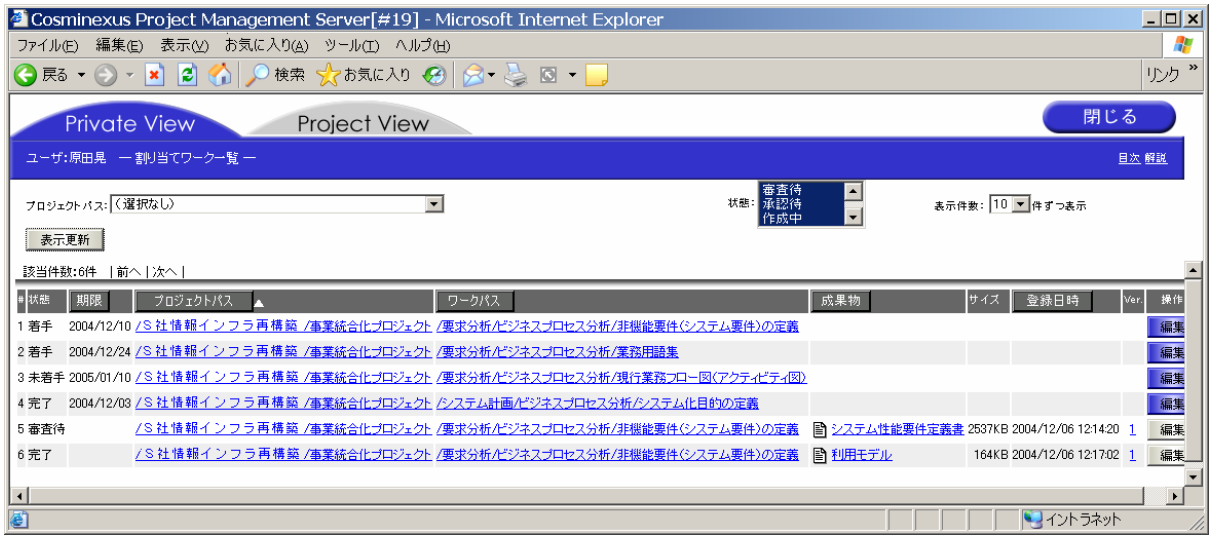

図 4.4 private view の表示例

(b)project view

プロナビ WBS と各ワークの成果物情報と，そこで利用する知識情報を示す画面である．画面例を図 4.5 に示す．

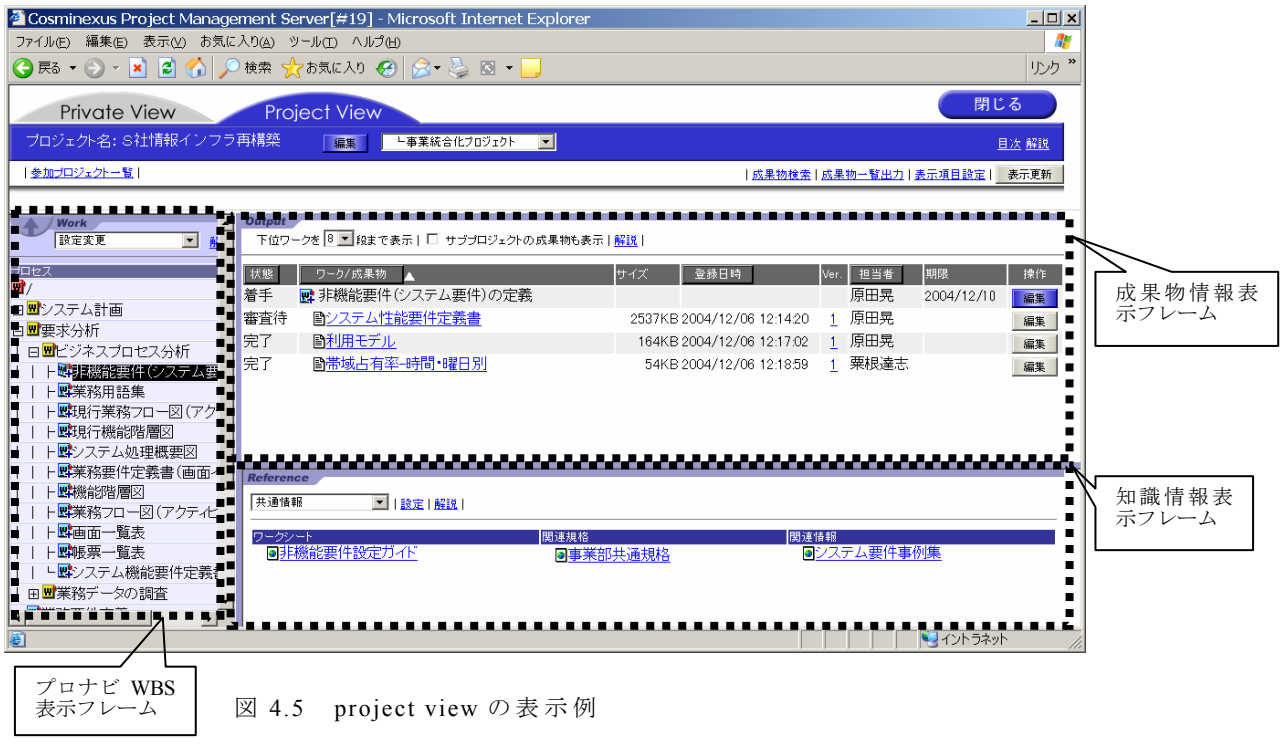

図 4.5 project view の表示例

画面左側のプロナビ WBS 表示フレームで見たいワークを選択すると，画面右上半面の成果物情報表示フレームに，そのワークの成果物ファイルの一覧，担当者，期限，進捗状況を表す状態が表示される．更に，一覧の中から成果物

ファイルを選択すると成果物として登録されたファイルの内容を参照できる．画面右下半面の知識情報表示フレームには,前提ワークの成果物一覧や知識情報一覧が表示される．その中から選択すると，該当するファイルの内容を参照できる．

例えば図 4.5 で，プロナビ WBS 表示フレームで「非機能要件（システム要件）の定義」を選択した場合，成果物情報表示フレームに，担当者は原田，期限は 2004 年 12 月 10 日で，着手済であることが示される．更に，非機能要件（システム要件）の定義は，システム要件定義書，利用モデル，帯域占有率―時間曜日別の 3 つの成果物ファイルから構成され,それぞれの担当者名と状態が示されている．また，参照情報表示フレームには，非機能要件（システム要件）の定義を行うときに利用する知識情報として、非機能要件設定ガイド，事業部共通規格，システム要件事例集が表示されている．

(3)進捗状況の確認

プロジェクト管理者は, project view の成果物情報表示フレームに表示されたワークや成果物ファイルの状態から進捗状況を確認する．また，必要があれば成果物ファイルそのものの内容を参照して，進捗状況や完成度を確認できる．

(4)To Do 管理

開発者は，クライアント PC に表示された private view 若しくは project view を参照し，自分に割り当てられたワークを確認する．これを"To Do 管理"と呼ぶ．また，project view を参照し，所属しているプロジェクトの計画や進捗状況を確認する．

(5)成果物の作成と To See 管理

開発者は private view 若しくは project view の成果物情報表示フレームに表示された一覧から自分の担当する成果物ファイルを選択しクライアント PC にダウンロードし，成果物の作成を進める．更新の終わった最新の成果物ファイルはサーバに送信され，成果物 DB に格納される．このとき，成果物ファイルのバージョン番号を 1 更新するが，バージョン番号により成果物ファイルの版管理がされる．作成が完了すると，成果物や成果物ファイルの状態を作成完に変更する．

成果物を作成する際, project view の知識情報表示フレームに表示された前提ワークの成果物一覧や知識情報から利用する資料を選択し，内容を参照する．

これを"To See 管理"と呼ぶ.

"To Do 管理"により，成果物の着手順序や作成期限が示され,"To See 管理"により，記載内容や記載形式の例や指針が示されるため，作業するときにナビゲーションがなされると考えることができる.

(6)承認機能

プロジェクト管理者は，成果物情報表示フレームに表示された一覧から，承認を依頼された成果物ファイルを選択して内容を確認し承認をする．また，この結果を状態に反映する.

4.4 評価

これまでのプロナビの適用状況と，適用効果について述べる.

4.4.1 適用実績

図 4.6 で適用プロジェクトの総数を示すように，2000 年 3 月からプロナビの適用を開始し，2005 年 5 月時点で，累計で約 2000 のプロジェクトがプロナビを適用してきている．また，常時，約 200 のプロジェクトがプロナビを適用中であり，プロナビは社内の標準的なプロジェクト管理システムとなっている．それらのプロジェクトから登録された設計ドキュメントのうち著作権を持つ約 5 万件はナレッジとして再利用されており，適切なアクセス管理の下に，社内の誰からも利用が可能であり，設計作業の効率化と高品質化に貢献している.

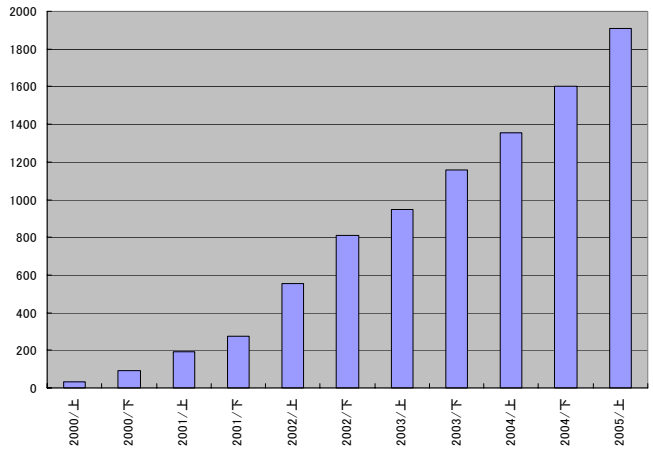

図 4.6 プロナビ適用プロジェクトの総数

4.4.2 プロナビを利用したプロジェクト管理の方法

プロナビによるプロジェクト管理の具体的な方法を，プロナビを利用しない

従来手法によるプロジェクト管理と対比して，表 4.1 にて説明する．

表 4.1　プロナビを利用したプロジェクト管理の方法

	従来手法によるプロジェクト管理	プロナビによるプロジェクト管理
管理者		
1)プロジェクト計画作成	・組織内に存在する WBS 作成標準や参考書を参考にして WBS モデルを作成する．作成された WBS モデルのワーク毎に担当者，開始日，終了日等を決めガントチャートを作成する． ・標準の WBS モデルが用意されていないので，WBS モデルの作成に手間がかかる．作成された WBS モデルの基本構造の統一を図ることが難しい．	・標準プロナビ WBS の中からプロジェクトに適したものを選択し，カスタマイズし個別プロナビ WBS を作成する．個別プロナビ WBS のワーク毎に，担当者，開始日，終了日，前提ワーク，参照する共通知識等の情報をプロナビ WBS に付加し，プロジェクト情報 DB に格納する． ・WBS モデルの作成が容易である．，作成された WBS モデルの基本構造は，管理者によらず，統一される．
2)進捗状況の把握	・定期的に開発者から進捗状況の報告を受けガントチャートに実績を記入する．事実と異なった報告のなされることや，報告者側の負荷が高くなる可能性がある ・成果物の完成度まで把握することができない．	・project view を参照してワークの進捗状況を把握する．必要に応じて成果物 DB に格納されている成果物ファイルの内容を直接，確認する．報告のための担当者の負荷が少ない． ・成果物の内容まで踏み込むことができるので，正確かつ詳細に進捗状況の把握ができる．
3)成果物の承認	・送付されてきた成果物の内容を確認して承認を行う．作成者は成果物が戻ってきて初めて承認結果を確認できるので，タイムラグが発生しやすい． ・成果物の受取や送付というアクションが必要になる．	・成果物 DB に格納されている成果物の内容を確認して承認を行い，結果をプロナビ WBS に反映させる．作成者は結果をすぐに知ることができ，タイムラグが発生しない． ・成果物の受取や送付というアクションは不要である．
4)成果物の管理	・電子的ファイル若しくは印刷物にして，決められたデータベースか書棚に保存し，プロジェクト内で共有できるように管理する． ・常に最新の成果物を管理するための負荷が大きく，かつ，版管理が不十分になりやすいので，最新情報の伝達が洩れることが多い．	・成果物ファイルとしてバージョン番号が付加され，プロナビの成果物 DB に格納されるので，プロジェクト内で常に最新の成果物の共有を実現でき，また，そのための負荷が殆どかからない．

表 4.1 続き

	従来手法によるプロジェクト管理	プロナビによるプロジェクト管理
開発者		
1)割当てられた作業及びスケジュールの確認	・ガントチャートを参照して確認する. ガントチャートは開発者毎でなく, ワーク毎になっているので, 割り当てられた作業とスケジュールを確認するには, 全てのワークをチェックする必要がある.	・private view を表示させるだけで, 割り当てられた全ての作業とスケジュールを確認することができる. ・project view を表示させることで,プロジェクト全体と自分に割り当てられた作業の関係を把握できる.
2)成果物作成	・前提ワークの成果物や知識情報を利用し, 成果物を作成する. 成果物はプロジェクトで管理しているデータベースや書棚から捜し出す. 知識情報は標準やワークシートのようにカテゴリー別に保存されており, そこから必要なものを捜し出す. このために必要な資料を捜すための負荷が大きく, 時には見つからない場合が起こる.	・project view を表示された前提ワークの成果物一覧や知識情報一覧から, 適したものを選択して内容を参照する. 表示される資料は, プロナビ WBS のワークと関連付けられており, 適したものを捜しだす負荷は非常に軽い. ・成果物の作成にあたっては, 知識情報の活用が促進され, 成果物の均等な品質を確保し易い.
3)成果物の承認依頼	・作成の完了した成果物を承認者に送付する. 承認者から戻ってきた成果物に付加された承認者のサインにより, 承認されたことを知ることができる. したがって, 承認結果をすぐに知ることができない. ・成果物の送付,受取というアクションが必要である.	・承認者にメール等で承認の依頼をする. 承認者は成果物 DB から該当の成果物を表示させ内容を確認し承認をし, 結果をプロナビ WBS に反映する. したがって, private view 若しくは project view を見ることで,すぐに承認結果を知ることができる. ・成果物の送付,受取というアクションは不要である.
4)進捗状況の報告	・定期的に進捗状況をプロジェクト管理者に報告する. 作業の完了した時点での報告でないために, 過去に完了した作業や進行中の作業の進捗状況を整理して報告する必要があり, 負荷がかかる.	・開発者は, ワークの着手, 作成完了, 承認の都度, private view 若しくは project view の該当するワークに, その事象を反映する. そのために, 定期的に進捗状況を整理して報告する必要がなく負荷が軽い.

4.4.3 適用評価

(1)アンケート結果

社内でプロナビを利用した 100 プロジェクトの管理者に対してのアンケートにより満足状況の評価を行った. 109 名から回答があり, 95%の人が満足していることが確認できた. また, 成果物等の情報の一元管理に最も満足したと回答した人が 34%, 分散した複数の開発場所から利用できることに最も満足した

と回答した人が 33%であった．即ち，プロジェクト計画，成果物，組織共通の知識，プロジェクト進捗状況等の情報が，プロナビ WBS により相互に関連付けられて一元管理され，それらをプロジェクトの誰もが参照や利用できることに満足していると回答した人が 2/3 を超えている．

(2)ヒアリング結果

アンケートとは別に，プロナビを利用した公共分野，産業流通分野，その他からの計 7 プロジェクトに対し適用効果と改善要望についてヒアリングを行った．それぞれ表 4.2，表 4.3 に纏める．

表 4.2 プロナビの適用効果

評価項目	効果
開発作業の高効率化	標準プロナビ WBS が用意されているので個別プロナビ WBS を簡単に作成できた．
	他の担当者が作成した最新版の成果物を簡単に参照できるので，仕様の確認が簡単であり，作業ミスがなくなった．
	作業手順，記述例等の知識情報を参照しながら作業をできるので，経験の浅い担当者でも効率的に作業を進めることができた．
	他の作業者の成果物を流用できるので効率があがった．
成果物の管理	最新版の成果物を即時に入手できた．
	版管理が行われているので，版の違いによる作業ミスがなくなった．
	目的の成果物を簡単に捜し出せるようになった．
情報の共有化	一つのプロナビのサーバに，物理的に離れている複数の開発拠点がネットワークで接続されており，プロジェクトのメンバ全員が常に同じ情報を即時に共有できた．
	プロジェクト基準書，開発計画書のようなプロジェクトの方針のプロジェクト内への周知が簡単になった．
進捗状況の把握	進捗状況の把握や成果物の内容確認等，煩雑になりがちな作業を簡略化することができた．
	直接，成果物の状態や内容を確認できるので，進捗会議での報告の裏づけを簡単にとれるようになった．
	成果物のチェックが簡単に行えた．

表 4.3 プロナビへの改善要望

改善要望事項	内容
成果物作成の進捗の集計機能	開発する機能毎に，その機能に関連した成果物の各状態の数を集計する機能があると，プロジェクト管理がより効率的になる．現在は，個々の成果物の状況から人手で集計している．
フォルダ単位の操作機能	Windows のエクスプローラと同様のフォルダ単位の操作が可能になると，大量の成果物や知識情報の参照や利用が，より簡単になる．現在はファイル単位での登録，参照，コピーしかできない．
プロナビ利用のベストプラクティスの収集	プロナビ利用のベストプラクティスを収集し，後続のプロジェクトに公開する．これにより，効率的なプロナビの利用が拡大していく．

4.4.4 分析・考察

プロナビの目的は，プロジェクトの工程，作業，成果物，参考資料等の情報を WBS モデルにより相互に関連付けし，一元管理することで，1)プロジェクト計画時の工程，作業，成果物の明確化，2)プロジェクト進捗状況の把握，3)プロジェクトメンバ間での成果物の共有，4)規則，標準，ワークシート等の組織に蓄積された知識の活用，5)開発プロセス及び作業の標準化を実現し，多数のプロジェクトメンバが複数の開発拠点に分散している大規模なエンタープライズ系ソフトウェアの開発プロジェクトでのプロジェクト管理の効率化に図ることであった．3), 4)については，アンケート結果からはプロジェクト計画，成果物，組織共通の知識，プロジェクト進捗状況等の情報が，一元管理され，プロジェクトの誰もが参照や利用できることに満足していると回答した人が 2/3 を超えており，これらを実現できていると評価できる．1), 2), 5)についても，プロジェクトへのヒアリング結果から実現されていると評価できる．また，プロナビは平成 12 年の適用開始以来，6 年間で 2000 を超えるプロジェクトで利用されてきており，現在も常時，200 を超えるプロジェクトで使い続けられているという実績からも，プロジェクト管理を支援する有用なシステムであると評価できる．

4.5 関連研究

一般のプロジェクト管理システムは種々提案され，製品化されている．その代表的なものである Microsoft Project は，ガントチャートを元にして，プロセス定義と進捗管理，構成管理などを含む成果物管理を行うことができる[55]．ProcessDirector は，ワークフローを元にして，Microsoft Project と同様の機能の他，情報共有機能等も有している[46]．プロナビは，プロジェクト管理の対象を主に大規模ソフトウェアに限定しており，WBS モデルを前提として各種の支援，管理作業を行っている．例えば，SLCP－JCF98 に準拠した標準プロナビ WBS を用意し，個別に変更可能な部分のみカスタマイズを許すことにより，プロセスの標準化を図っている．Microsoft Project 等のツールには，そのような機能はない．

大規模なソフトウェア開発のプロジェクト・プロダクト管理の例としては，PMOffice Enterprise が知られている[25]．このシステムは,Microsoft project を用

いた WBS モデルに基づくプロジェクト管理を，独自のデータベースを使って効率的に行うことを目指している．しかし，実際に大規模なソフトウェア開発のプロジェクトのプロジェクト管理に有効に働くか，また，プロジェクトメンバに，どの程度有用な機能を提供できるかについては，詳しい報告はない．

品質管理と進捗管理で用いる資料や帳票を管理するプロジェクト管理システムについても研究があるが[57]，実際のプロジェクトへ適用した場合の評価についての報告はない．

4.6 結論

本章では，プロジェクトの工程，作業，成果物，規則，標準，手順，ワークシート，過去のベストプラクティスのような組織に蓄積された知識情報を相互に関連付けし，一元管理するための WBS モデルの提案と，その WBS モデルを利用したプロジェクト管理システム「プロナビ」の開発を行った．4.4 節に示すように，プロナビは，プロジェクト管理を支援するシステムとして有効であることを確認できた．

一方，進捗状況の詳細な集計や，より細かな成果物への操作，改善要望もいくつか出ており，より優れたプロジェクト管理システムとなるように，改良を進めている．

第 5 章　プロジェクト管理システム「プロナビ」の進捗管理機能強化

5.1　緒言

第 4 章において，プロジェクト管理システム「プロナビ」の開発と評価を報告した．

プロナビの目的は，プロジェクトの工程，作業，成果物，参考資料等の情報を WBS モデルにより相互に関連付けし，一元管理することで，1)プロジェクト計画時の工程，作業，成果物の明確化，2)プロジェクト進捗状況の把握，3)プロジェクトメンバ間での成果物の共有，4)規則，標準，ワークシート等の組織に蓄積された知識の活用，5)開発プロセス及び作業の標準化を実現し，多数のプロジェクトメンバが複数の開発拠点に分散している大規模なエンタープライズ系ソフトウェアの開発プロジェクトでのプロジェクト管理の効率化に図ることであった．

これらについて，実際にプロナビを利用したプロジェクト管理者に対し，アンケートやヒアリングを実施し，プロジェクト管理を支援する有用なシステムであることが確認できた．

一方，成果物作成の進捗管理において，現在は，個々の成果物の状況から人手で進捗を集計していて負荷がかかるので，プロナビにより進捗情報を自動的に収集する機能が欲しいという要望がでていた．

プロナビを適用しない進捗管理では,ガントチャートを用いることが多いが,その場合,ガントチャートで報告される進捗状況と,実際の成果物の作成状況との間で乖離が発生することが多い．

プロナビにより，成果物作成の進捗の実態に基づいた進捗情報を自動的に収集できると,恣意性を排除した進捗情報をタイムリーに把握できることになり,強力なプロジェクト管理システムになると考えられる．

5.2　プロナビ WBS モデルの変更

5.2.1　基本方針

成果物の実態から進捗状況を機械的に把握するには,次の 3 点が必要である．

・プロナビ WBS の基本構造が，進捗の管理単位と一致していること

・成果物の粒度が適切に設定されていること

・成果物の状態を正しく管理できること

これらを，実現できるように，プロナビ WBS の基本構造を変更する．

5.2.2 プロナビ WBS の基本構造

現在のプロナビ標準 WBS を図 5.1 に示す．

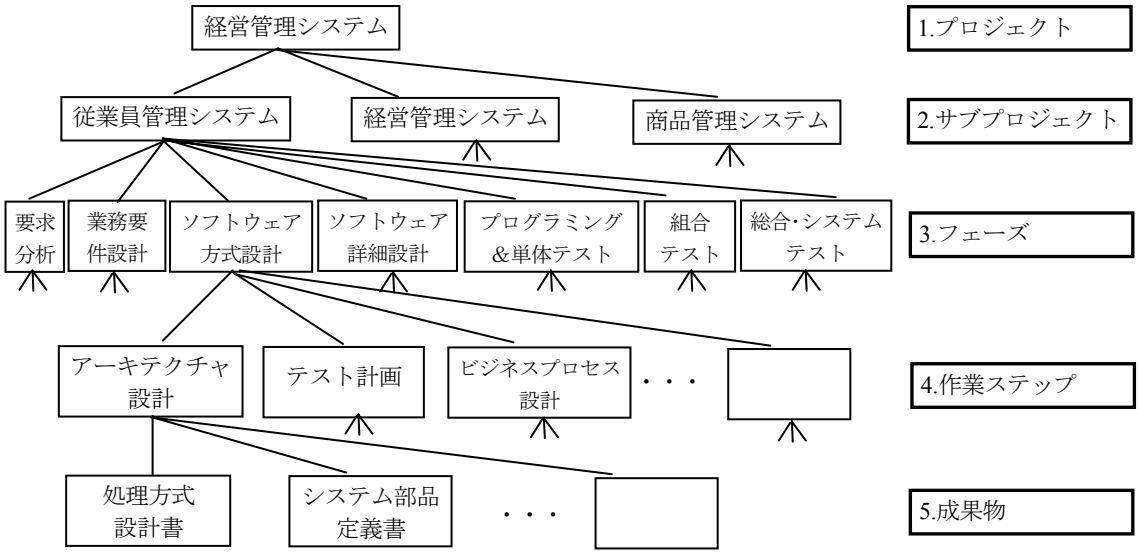

図 5.1 標準プロナビ WBS

個々のプロジェクトに標準プロナビ WBS を適用する場合，ワークの過不足が発生する場合がある．そこで，標準プロナビ WBS に対し，必要なワークの追加，不要なワークの削除，ワーク名の変更等を行い，標準プロナビ WBS を自プロジェクトに適用できるようにした個別プロナビ WBS を作成する．したがって,各プロジェクトの個別プロナビ WBS は，一定の基本構造をもったものとなる．

実際の個別プロナビ WBS が，どのような構造になっているかについて，プロナビを利用した 50 のプロジェクトを調査した．その結果を表 5.1 に示す．

表 5.1 プロナビ個別 WBS の構造パターン

パターン名	プロナビ WBS 構造パターン	プロジェクト数	割合(%)
パターン 1	工程－作業－成果物－機能	29	58%
パターン 2	工程－成果物－機能	16	32%
パターン 3	工程－作業－機能－成果物	2	4%
パターン 4	工程－機能－成果物	1	2%
パターン 5	工程－作業－成果物	1	2%
パターン 6	工程－成果物	1	2%

表 5.1 から，個別プロナビ WBS には，図 5.1 の標準プロナビ WBS にない「機能」という階層が表れていることがわかる．

機能と成果物の観点から表 5.1 を分析すると，成果物が機能よりも上位に位置づけられている構造パターンは，パターン 1 とパターン 2 であり，全体の 90%である．一方，機能が成果物よりも上位に位置づけられているのは，わずか 6%である．

5.2.3 進捗管理に適した WBS モデル

進捗情報の集計では，最下位に位置する成果物の進捗を積上げて上位の階層で集計するので，WBS の構造が重要である．WBS の構造が進捗の管理単位に適さない場合，その構造に従い，機械的に進捗を集計しても意味のある情報にはならない．

一般に，システム化する業務を，概念や機能の独立性を基に，サブシステム，機能単位に分割する．この体系に基づき，開発チームや開発者を割当てる．進捗管理は，開発チーム単位に実施するので，進捗の管理単位は，機能単位に行うのが妥当であると言える．

次に，WBS 上での，成果物と機能の階層の位置と，進捗状況の把握の関係について評価する．

図 5.2 は，パターン 1, パターン 2 のように，機能が成果物より下位に位置づけられた WBS の構造である．

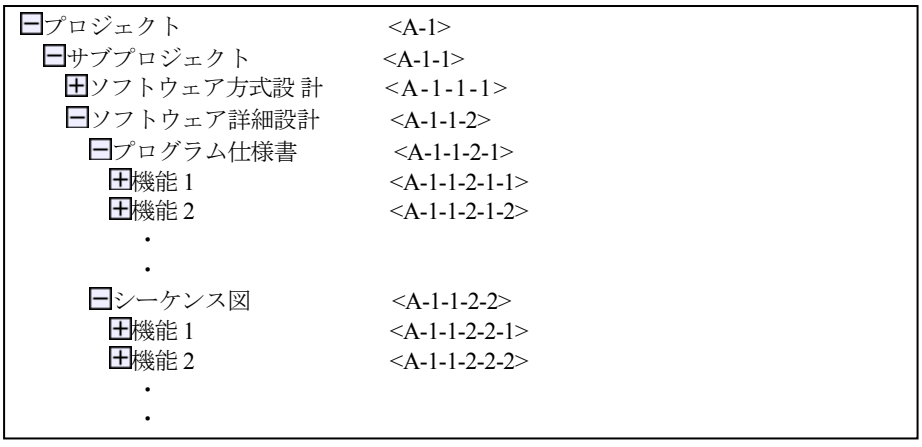

図 5.2 プロナビ WBS の例（機能が成果物より下位の場合）

この WBS の場合，機能 1 の進捗状況を把握するには，プログラム仕様書<A-1-1-2-1>の下位の機能 1<A-1-1-2-1-1>と，シーケンス図<A-1-1-2-2>の下位の

機能 1<A-1-1-2-2-1>の両方の進捗から集計する必要がある．したがって，図 5.2 のプロナビ WBS の構造は,進捗情報の自動集計に適していないと考えられる．

一方，表 5.1 のパターン 3，パターン 4 の WBS では,機能が成果物の上位に位置している．この WBS の構造を整理すると図 5.3 になる．

図 5.3 プロナビ WBS の例（機能が成果物より上位の場合）

図 5.3 では，機能 1<B-1-1-2-1>の階層の下位に，プログラム仕様書<B-1-1-2-1-1>とシーケンス図<B-1-1-2-1-2>の成果物が位置づけられている．これらの成果物の進捗情報が機能 1<B-1-1-2-1>の進捗情報として集計されるので，機能 1<B-1-1-2-1>を見るだけで進捗がわかる．したがって，図 5.3 のプロナビ WBS の構造は，進捗管理に適していると考えられる．

5.2.4 成果物の粒度

成果物の作成状況から進捗を把握できるためには，適切に分割されている必要がある．

分割が適切でない例として，画面仕様書という成果物がシステムで 1 個しかない場合を想定してみる．ある機能の画面仕様の作成は完了しているが，別のある機能の画面仕様の作成が完了していない場合，画面仕様書の作成状態を見るだけでは，どの機能が遅れているか，機械的には把握できない．しかし，画面仕様書が機能単位に，別の成果物として分割されていれば，どの機能の画面仕様書が完了して，どの機能の画面仕様書が完了していないかが，機械的に把握できる．

図 5.3 のようなプロナビ WBS の構造にすることで,成果物は機能単位の粒度に分割されることになり，成果物の作成状況から機械的に進捗を把握しやすくなる．

5.2.5 成果物の状態管理

プロナビでの進捗管理は，成果物と WBS の各階層の状態を集計した値で行う．したがって，状態を正しく管理できることが必要である．成果物が承認されていないのに,“承認済”という状態が不当に設定されることを防ぐ仕掛が必要となる．この実現のために，責任をもつ人だけに状態を変更できる権限を与えるようにアクセス制御を行う．例えば，プロジェクト管理者だけが,“承認済”という状態を設定できる仕掛があれば，成果物は必ずプロジェクト管理者が承認したということが保証でき，正確な進捗管理が可能になる．

5.3 実現方式

5.3.1 概要

5.2 節で説明したことを実現するために，プロナビに次の 3 つの変更及び機能追加を行う．

・機能の階層を組込んだプロナビ WBS のモデル化

・成果物の状態管理機能

・進捗情報の集計とレポート機能

5.3.2 機能の階層を組込んだプロナビ WBS のモデル化

ソフトウェアの開発規模の見積では，複数の機能に分割し，機能単位に開発規模を見積る方が，見積も容易で，かつ精度も高くなると考えられる．また，機能の分割も 1 階層よりも，大機能，中機能の 2 階層に分ける方が，更にそれが徹底される．中機能毎に開発規模を表 5.2 のように見積した例を基に，図 5.4 のようにプロナビ WBS を作成する．このプロナビ WBS では成果物の上位に機能を位置づけており成果物が適切な粒度に分けられることを期待できる．

表 5.2 機能一覧

大機能	中機能	FP
大機能 1	中機能 11	20
		25
		15
	中機能 12	20
		30
	中機能 13	25
		20

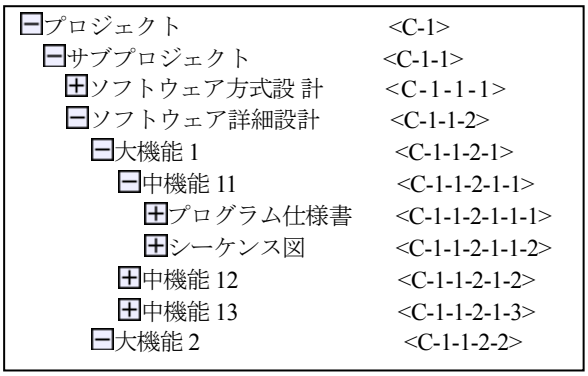

図 5.4 機能の階層を組込んだプロナビ WBS モデル

5.3.3 成果物，WBS 階層の状態管理

(1)表 5.3 に示すように，プロナビの成果物の状態定義を行う．

表 5.3 成果物の状態定義

状態	定義	更新者
着手	成果物の作成に着手した状態	担当者
作成完了	成果物の作成が完了した状況	担当者
QA 承認済	品質保証部門(QA)が承認した状態	品質保証部門
PM 承認済	プロジェクトマネージャ(PM)が承認した状態	PM
顧客承認済	顧客が承認した状態	PM

状態管理で重要なことは，客観的に状態を判断できること，正しく状態を管理できることである．表 5.3 に示す成果物の状態は，客観的に判断可能であり，プロジェクトメンバの間で認識を共有しやすい．

成果物の状態の更新のうち，QA 承認済，PM 承認済，顧客承認済の 3 つは，表 5.3 の更新者に示すように責任のある者のみに更新を制限しており，状態が正しいことが保証される．

(2)表 5.4 に示すように，WBS 階層の状態定義を行う．

表 5.4 WBS 階層の状態定義

状態	定義	更新者
未完了	当該階層以下の成果物が完了していない状態	PM
完了	当該階層以下の全成果物が完了した状態	PM

表 5.4 の状態についても，客観的に判断可能である．更新者もプロジェクト管理者のみに制限するため，状態が正しいことが保証される．

(1),(2)により，成果物及び WBS 階層の状態を，正しく管理することができる．

5.3.4 進捗情報レポート機能

WBS の各階層の情報と成果物の状態より，進捗情報を集計し，レポート出力する機能を組込む．

表 5.5 は，サブプロジェクト 1 の進捗情報をレポートした例である．WBS 階層の情報と成果物の状態を示している．

(1)レポートの項目の説明

・FP：WBS の当該階層以下に含まれる機能の FP の合計．

この FP 値は，表 5.2 の機能一覧から WBS を作成するときに設定した値である．

・進捗：当該階層以下に含まれる完了した機能の FP のみを集計した値．

表 5.5 の中機能のような機能階層の進捗は，当該機能より下位に存在する全成果物が完了して初めて当該機能の FP を計上する．未完了の成果物が 1 つでも存在する場合，当該機能の進捗は 0 とする．

・階層の状態：下位の階層に存在する全成果物が完了したか否かを示す．

・担当：各機能の担当者，若しくは当該機能を担当する開発チームの責任者．

・成果物予定数：各機能における作成予定の成果物数．

・成果物の状態：機能の下位に存在する成果物の状態を示している．

表 5.5 進捗状況のレポート出力例

工程	大機能	中機能	FP	進捗	階層の状態	担当	成果物予定数	成果物の状態				
								顧客承認済	PM 承認済	QA 承認済	作成完了	着手
ソフト基本設計	－	－	－	－	完了	A	50	50	50	50	50	50
ソフト詳細設計	－	－	300	180	未完了	B	80	70	73	78	78	78
	大機能 1	－	180	60	未完了	C	50	40	43	48	50	50
		中機能 11	60	60	完了	D	15	15	15	15	15	15
		中機能 12	65	0	未完了	E	25	20	20	25	25	25
		中機能 13	55	0	未完了	F	10	5	8	8	8	8
	大機能 2	－	120	120	完了	G	30	30	30	30	30	30

(2)レポートの見方

(a)全体若しくは WBS 階層単位での進捗の把握

WBS 階層に対する“進捗”の項目を見る.

表 5.5 のソフトウェア詳細設計では，全機能の FP が 240 に対し進捗が 180 であり，75%の進捗になる.

(b)機能単位の進捗の把握

成果物の予定数と，成果物の状態から評価する.

表 5.5 の中機能 13 は，未着手の成果物が 2 件残っており，他と比較して進捗が遅れていることが分かる．また，各状態での数を見ることで，成果物がどの状態で止まっているかを確認できる.

中機能 12 の場合，QA 承認済が 25 件であるのに対し，PM 承認済が 20 件となっており，PM のところで 5 個の成果物の承認が滞っていることが分かる.

このように，最新の進捗情報をいつでも取得可能であり，問題部位の特定が容易である.

5.4 評価

プロナビの進捗管理機能の強化により，次の効果を得ることが確認できた.

(1)進捗情報の集計の機械化により，タイムリーな進捗の把握が可能となる.

(2)恣意性を排除でき，成果物の実態に基づいた進捗情報を得ることができる.

(3)機能単位に分割した成果物の状態を把握することで，どの機能の成果物の作成が遅延しているかを機械的に特定できる.

5.5 結論

プロナビ WBS のモデルを，進捗の把握の自動化という観点から再検討し，機能の階層を組込んだ WBS モデルを採用することで，プロナビによる成果物の実態に基づいた進捗の自動収集を実現することができた．また，収集した進捗情報は恣意性を排したものであり，プロジェクトの進捗を管理する上で非常に有用な情報であると言える．これにより，プロナビに対する改善要望に応えることができ，プロジェクト管理を支援する一層，有効なシステムに強化できたと判断できる.

第 6 章　QFD を応用した機能仕様の早期確定技法

6.1　緒言

エンタープライズ系ソフトウェアの受託開発ではプロジェクトの約 40%が，当初設定した予算や納期に収めることができない，テスト段階や顧客への納品後に品質上の重要な欠陥が発見される等の，いわゆる失敗プロジェクトに終わると言われている[30],[31].

これらの主たる原因は見積誤りによるコスト超過，不適切なプロジェクト管理の２つである．これは，昨今の顧客をとりまくビジネス環境の変化の速さ，IT の進歩の速さに起因することはいうまでもないが，より本質的な問題として，ソフトウェアの性質に起因するところが大きいのである．ソフトウェアは無形物であるため，プロジェクト初期の段階では仕様を明確にしにくく，プロジェクト進行とともに少しずつ仕様を明確にしていかなければならないという性質を持つ．そのことにより，見積誤りの誘発や，プロジェクト管理の困難さを引き起こしているのである．

これらの課題を解決するには，下記のような方策をとることが必要と考えられている．

①ソフトウェアの仕様や制約,プロジェクトの作業内容,顧客との作業分担等，コストを左右する部分の要件や前提条件を見積段階で明確にすること．

②その上で，これらの変更を常に追跡し，コストやスケジュールへの影響を把握し，その影響度に応じた適切な措置をとること．

追跡すべき項目として特に重要なのがソフトウェアの仕様である．仕様変更が多発すると必然的に開発作業量も増加し，コスト増に直結する．開発規模の実績が見積時点の 2 倍以上に達することもあり，ソフトウェアの仕様の確定状況を管理することがソフトウェア受託開発においては極めて重要である．

この課題を解決するために，定量化，可視化および重点化指向といった QFD(Quality Function Deployment：品質機能展開)[4],[5]の特徴を応用した仕様発散防止 QFD と呼ぶ技法を提案する．

6.2　QFD の概要

QFD では，顧客要求を技術的な特性に変換するために，品質表と呼ばれる 2 元表を用いる．顧客要求を整理したものを要求品質と呼び，要求品質から変換

された技術的特性のことを品質要素と呼ぶ．品質要素は，品質を評価する尺度となりうる要素を指し，具体的には顧客要求を実現するための機能である．また，要求品質項目毎に，重要度とセールスポイント（他社競合比較）を設定したものを企画品質と呼ぶ．

図 6.1 に示すように，Y 軸に要求品質，重要度，セールスポイントを設定し，品質要素を X 軸に設定する．次に，X 軸と Y 軸との交差部分に XY 軸の各項目の関連度を設定していくことで，各要素の因果関係を把握する．

この結果，要求品質項目間の優先度が企画品質として決定され，更に，品質要素間の優先度が決定される．この優先度に応じて実現する機能を選択し，設計，製造を進めていけばよい．

QFD を適用することにより，次の効果を期待できる．

・顧客要求を技術者の言葉である品質要素に翻訳することにより，要求定義の設計を容易にする．

・設計に採用すべき技術的手段の決定過程が，定量化，明確化，可視化できる．

・顧客要求を達成するための計画を，論理的に展開し，品質表として可視化することにより，情報の共有を容易にし，更に，知識の蓄積を容易にする．

・広い範囲の対象の中から，顧客要求の重点化が可能である．

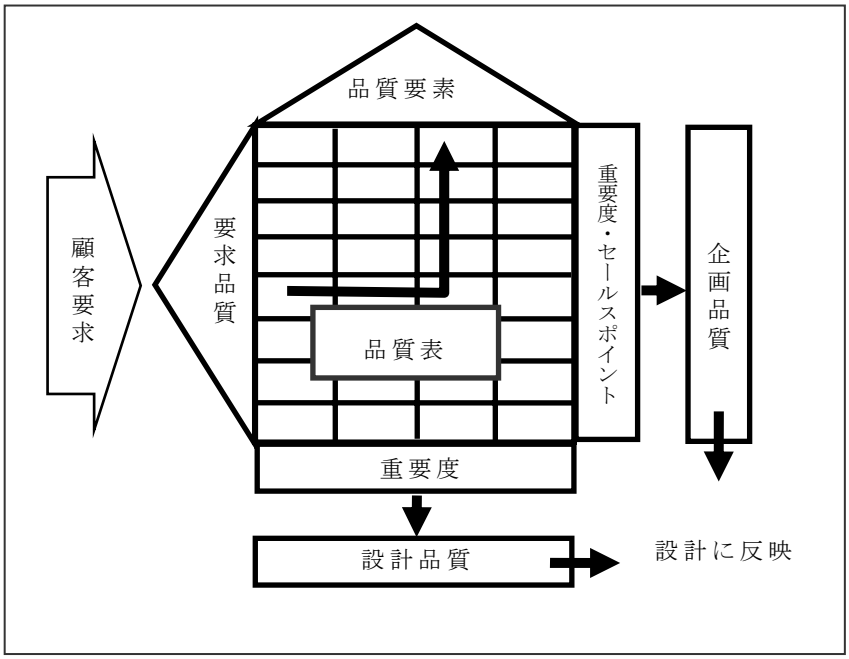

図 6.1 QFD の概要図

6.3 仕様発散防止 QFD

6.3.1 QFD の有効性

QFD は，顧客の声を設計，製造といった技術的手段へ変換するための適切なプロセスを提供する技法である．QFD は，定量的な評価手段により決定過程を明確化，可視化することで，定型化による意思伝達手段や重点指向，網羅性を実現する．

一方，ソフトウェア開発のプロジェクト管理に，定量化，可視化，重点指向の方法を採用するのは有効である.具体的には,ソフトウェアの備えるべき各々の機能について，仕様確定状況を定量的に監視し，機能の重要度や複雑度を加味して，対策の要/不要の判断や対策すべき機能の順序付けを行うことである．ソフトウェアの仕様や制約，プロジェクトの作業内容，顧客との作業分担等の内容を数値で定量化することは課題解決への第一歩である．これを実現する技法として研究したものが，QFD を応用した仕様発散防止 QFD である．

失敗プロジェクトでは，どこに問題が潜んでいるか，またそれがどのような要因によるものかをプロジェクトリーダクラスの人が把握できていないことが多い．また，失敗プロジェクトでも，全ての機能に問題があるのでなく，実際には一部の仕様が安定せず,その影響が広範囲に及んでいるような場合が多い．仕様発散防止 QFD を利用することで，これらの課題を解決，若しくは改善することが期待できる．

6.3.2 仕様発散防止 QFD の概要

仕様発散防止 QFD では,仕様確定の遅延への対策の必要がある機能の特定や,対策の優先度を決めるために，各機能に対し危険度と呼ぶ指標を定義する．

危険度は，機能のニーズを表す機能重要度と，機能の実装の難しさやシステム全体に及ぼす影響を表す機能複雑度と呼ぶ指標から,次の手順で算出される．

①機能一覧の作成

顧客要求から，QFD の要求品質に該当する機能一覧を作成する．

②機能重要度の設定

顧客ニーズを基に次の順序で機能重要度を設定する．

②-1 ユーザ一覧の作成

②-2 ユーザ重要度設定

②-3 ニーズ一覧の作成

②-4 ニーズ重要度の設定

②-5 機能×ニーズ突合せによる機能重要度の設定

③機能の複雑度の評価

機能の複雑度を，仕様確定遅延時や仕様変更発生時の全体への影響の観点から評価する．

④機能の危険度の評価

機能複雑度に，仕様の提示度合，問題解決状況を加えて評価する．

①，②，③はプロジェクトの開始時に実施する．④は全ての仕様が確定するまで，1 週間毎や 2 週間毎のように定期的に実施する．

6.3.3 機能重要度，機能複雑度，機能危険度の算出

(1)機能重要度

機能重要度はその機能に対し，顧客がどのようなニーズをもっているか，その顧客のどの程度の決定権を持っているかによって設定する．図 6.2，図 6.3 に示すように，次の順序で行う．

①ユーザ一覧の作成

システムには，エンドユーザ，システムユーザ，経営者等のさまざまなユーザがおり,顧客ニーズの評価を多面的な観点から行うために,ユーザを洗出し，一覧にする．

②ユーザ重要度の設定

洗い出したユーザ種別に対して，仕様決定権の強さ，システムを利用する人数規模，システムの利用頻度により評価を行い，ユーザの重要度を設定する．

③ニーズ一覧の作成

システムに対するコンセプトや期待効果等を一覧にする．

④ニーズ重要度の設定

ニーズに対し,それを必要としているユーザの重要度を基に，ニーズの重要度を設定する．

⑤機能×ニーズ突合せによる機能重要度の算出

機能に対するニーズ重要度から機能重要度を算出する．

機能重要度の大きい機能ほど，顧客からの注目度が高いと判断できる．

		ユーザ一覧						ニーズ重要度
		ユーザ1	ユーザ2	・・	ユーザn	・・	・・	
		u_1	u_2		u_n			
ニーズ一覧	ニーズ 1	a_{11}	a_{12}		a_{1n}			a_1
	ニーズ 2	a_{21}	a_{22}		a_{2n}			a_2
	・							
	ニーズ m	a_{m1}	a_{m2}		a_{mn}			a_m
	・							

u_n：ユーザ n の重要度．重要度が大きいほど，1～3 の範囲で値が大きくなる．

a_{mn}：ニーズ m に対するユーザ n の必要度合．必要度合いが大きいほど，1～3 の範囲で値が大きくなる．

a_m：ニーズmの重要度．$a_m=a_{m1}\times u_1 + \cdots + a_{mn}\times u_n + \cdots$

図 6.2 ニーズ重要度の算出

(2)機能複雑度の算出

各機能についての機能複雑度を，表 6.1 に示す 5 つの評価項目について評価して，図 6.3 に示すように算出する．

機能複雑度が大きいほど，設計，製造の難易度が高いと考えられる．

表 6.1　機能複雑度の評価項目一覧

評価項目	評価観点
機能特性	システム全体におけるその機能の重要性を評価する．
システム I/F 有無	システムインタフェースの有無を評価する． システムインタフェースの状況は開発の進行に大きく影響する．
接続マシンの新規性	接続先の他システムの状況を評価する．接続先が同時開発である場合，接続先の仕様変更や開発遅延が自システムに大きく影響する．
マンマシン I/F	画面やコマンド等のマンマシンインタフェースの状況を評価する．マンマシンインタフェースは仕様変更が起こりやすいことが知られている．
他機能との連携度合い	自システム内の他機能との連携度合を評価する．他機能との関係が強い機能は，早めに仕様を固めないと他の機能の仕様決定にも影響を与える．連携度合の「1 割」,「3 割」は感覚的なもので構わない．DB 項目を介して連携しているケース(A 機能で更新，B 機能で参照というような場合)では，その項目がフラグやステータスの類である場合には機能間の連携ありと判断する．この項目がデータそのものである場合は連携とはみなさない．

		ニーズ一覧					機能重要度	機能複雑度評価					機能複雑度
		ニーズ1	ニーズ2	・	ニーズn	・		評価項目1	評価項目2	評価項目3	評価項目4	評価項目5	
		a_1	a_2		a_n								
機能一覧	機能 1	b_{11}	b_{12}		b_{1n}		b_1	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	d_1
	機能 2	b_{21}	b_{22}		b_{2n}		b_2	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}	d_2
	・												
	機能 m	b_{m1}	b_{m2}		b_{mn}		b_m	c_{m1}	c_{m2}	c_{m3}	c_{m4}	c_{m5}	d_m
	・												

a_n：ニーズ n の重要度.

b_{mn}：機能 m とニーズ n の関連性．関連性が大きいほど，1～3 の範囲で値が大きくなる.

b_m：機能 m の重要度．$b_m = b_{m1} \times a_1 + \cdot\cdot b_{mn} \times a_n \cdot\cdot$

c_{m1}，c_{m2}，c_{m3}，c_{m4}，c_{m5}：機能 m に関する表 6.1 に示す 5 つの評価項目の評価値．1.0～2.0 の範囲であり，値が大きいほど複雑度が高くなる.

d_m：機能 m の複雑度．$d_m = b_m \times c_{m1} \times c_{m2} \times c_{m3} \times c_{m4} \times c_{m5}$

図 6.3 機能重要度，機能複雑度の算出

(3)機能危険度の算出

機能危険度は,機能複雑度に加えて,表 6.2 の刻々と変化する要因を評価し，図 6.4 に示すように，定期的に算出する.

		機能複雑度	危険度評価					
			第 1 回目評価			第 2 回目評価		
			業務仕様提示度合	懸案残存率	機能危険度	業務仕様提示度合	懸案残存率	機能危険度
機能一覧	機能 1	d_1	e^1_{11}	e^1_{12}	e^1_1	e^2_{11}	e^2_{12}	e^2_1
	機能 2	d_2	e^1_{21}	e^1_{22}	e^1_2	e^2_{21}	e^2_{22}	e^2_2
	・							
	機能 m	d_m	e^1_{m1}	e^1_{m2}	e^1_m	e^2_{m1}	e^2_{m2}	e^2_m
	・							

e^1_{m1}：第 1 回目評価における機能 m に関する業務仕様提示度合.
1～3 の値をとり，値が小さいほど詳細な仕様が提示されていることを示す.

e^1_{m2}：第 1 回目評価における機能 m に関する懸案残存率．0.0～100.0

e^1_m ：第 1 回目評価における機能 m の危険度．$e^1_m = d_m \times e^1_{m1} \times e^1_{m2}$

図 6.4 機能危険度の算出

機能危険度が大きいほど，仕様の確定が遅れていて，プロジェクト全体に対する影響が大きいと考えられる．

表 6.2　危険度の評価項目一覧

評価項目	評価内容
機能仕様提示度合	顧客から仕様がどの程度提示されているかを示す．
懸案残存率(%) (未解決懸案数/懸案数)	未解決懸案が多いほど，その機能の開発は遅延する．

6.3.4　仕様発散防止 QFD の活用方法

仕様発散防止 QFD のプロジェクト管理での活用について説明する．

機能複雑度や機能危険度が大きいという現象は，その機能に何らかの対策が必要であるという警告になるが，実際にどのような対策が必要になるかは，その時の状況により異なる．

活用例を挙げる．

(1)機能重要度，機能複雑度の活用方法

機能重要度，機能複雑度を評価することで，優先度の高い機能に的を絞ることができる．仕様の設計段階において，顧客から仕様の追加や変更の要求が発生し，当初の計画よりも仕様が増大する危険がある．このような場合，顧客の業務遂行に必要な機能に絞込むことが必要となるが，仕様発散防止 QFD を適用し，機能重要度と機能複雑度を評価することで，仕様の絞込みを体系的かつ定量的に実現できる．例えば，機能重要度，機能複雑度とも大きな機能は，開発開始時から優先的に取組む必要がある．また，機能重要度が小さく，機能複雑度の大きな機能は，縮小や削減の対象としやすい．このような方針は，定量的に評価した結果であるので，顧客の理解を得られやすい．

(2)機能危険度の活用方法

機能危険度から，仕様を早く詳細化すべき機能の把握や，仕様の収束していない機能の検出ができる．

(a)機能危険度が大きい場合

この機能に何らかの対策が必要であることを警告している．特に，開発者のスキルと現状体制の十分性に問題がある場合が多い．

(b)機能危険度が大きいまま推移している場合

危険度評価を複数回実施した場合,前回より上がっている,下げ幅が小さい,変化がないなどの機能が顕在化でき,原因の究明が必要である.体制,スキル,顧客等に何らかの問題がある場合が多い.

懸案数の推移（異常に増加，解決が非常に少ない等），進捗報告と比べて矛盾がないかにも注意が必要な場合が多い.

(c)機能一覧が作成できない場合

機能一覧が作成できない,機能複雑度や危険度を算出するための評価項目の評価ができない場合,プロジェクト管理ができていない,現状把握できていない可能性が多い．プロジェクト管理の強化が必要である.

また,プロジェクトの初期段階では，このような状況が発生しやすいが，機能一覧の作成，表 6.1 の評価項目，機能仕様提示度合，懸案状況の明確化は,プロジェクト管理を効率よく進めるのに必要な事項であり,早急に実施すべきである.

6.3.5 仕様発散防止 QFD の効果

仕様発散防止 QFD の特徴を次のように纏めることができる.

(1)開発規模増加の早期発見

プロジェクトの混乱につながる開発規模の増加を早期に発見できるため，開発工数，期間およびシステムの品質を高く保つことができる.

(2)プロジェクトの仕様遅れの監視

プロジェクト全体に影響の大きな仕様変更や仕様確定の遅れに対しては，危険度が高くなり，定量的に表現できるため，問題の早期発見をしやすい.

(3)顧客視点での機能重要度の定量化によりプロジェクト管理をサポート

仕様が膨大になった場合，顧客ニーズを反映した機能の優先順位付けが定量化，可視化できるので，プロジェクト管理で重点化するところの特定が容易である.

6.4 適用評価

(1)的確な対策の優先順位付け

仕様発散防止 QFD を適用した 30 件のプロジェクトについて，アンケートを

実施した．この結果,「機能複雑度や危険度は，その時点でプロジェクトの主な問題がどこに潜んでいるかを的確に示したと思うか」という問いに対し,「そう思う」と回答した割合が 90%に達した．

また，定められた納期を達成できなかったプロジェクトは 6%程度 30 件中 2 件しかなかった．

これらの事実から仕様発散防止 QFD が的確な対策優先順位付けに効果を発揮し，結果として納期遅延を防いでいると評価できよう．

(2)可視化

プロジェクト全体をいくつかの機能に分類し，それらの状況を A3 程度の大きさの資料に纏めることで，プロジェクト全体の状況把握が容易に行えるようになり，問題箇所の発見がスムーズに行えるようになった．

この点は特にプロジェクトマネージャや，より上位の管理者から高い評価を得ている．

(3)適用効果の小さかったケース

アンケートで「機能複雑度や危険度は，その時点でプロジェクトの主要な問題がどこに潜んでいるかを的確に示したと思うか」という問いに「そうは思わない」と回答したプロジェクトが 10%ある．

この原因を調査した結果，次の２つにパターン化できる．

・どの機能も危険度が高い場合

特定部位の対策でなく，全体的な根本対策が必要であることを示す．

・認識通りの結果である場合

スキルの高くない技術者，プロジェクトマネージャでも仕様発散防止

QFD を適用することで，同様の認識を持つことができ，効果は大きいと判断できると考えられる．

6.5 結論

本論文では，仕様発散防止 QFD を提案し，適用効果を評価した．

この結果，仕様発散防止 QFD は，定量化，可視化および重点化指向といった QFD の特徴を活かすことができ，曖昧な機能仕様を早期に確定する技法として，効果のあることを確認できた．

ソフトウェアは形が見えないために，顧客やプロジェクトメンバに定量的かつ客観的に説明することが難しい事柄が多いが，仕様発散防止 QFD のような考え方を取入れることができる分野が多いように思う．

今後もこの方法を発展させていくと共に，プロジェクト管理の定量化，可視化および重点化指向を進めて行きたい．

第 7 章　むすび

7.1　まとめ

本研究では，エンタープライズ系ソフトウェア開発のプロジェクトを成功させるために特に重要と考えられる，次の 3 つの技術の実現を目標とした．

目標 1：早い段階での明確な根拠に基づいた精度の高い見積技法

目標 2：大規模なプロジェクトのプロジェクト管理を幅広く支援するシステム

目標 3：機能仕様の追加，変更の多発を防止し，仕様を早期に確定する技法

そこで，これらの目標を達成するために，次の 4 つの研究を行った．

(1)ファンクションポイント法を応用した早期見積技法

(2)WBS に基づくプロジェクト管理システム「プロナビ」

(3)プロジェクト管理システム「プロナビ」の進捗管理機能強化

(4)QFD を応用した機能仕様の早期確定技法

(1)については，要素見積法と呼ぶ,画面上の 1 個の GUI ボタンや，1 回のファンクションキーを押すことで実行される具体的な処理 16 種類を要素機能とし，要素機能がいくつあるかでファンクションポイントを算出する技法を開発し，評価した．

IFPUG 法の適用が難しい開発の前段階で，要素見積法により見積を実施したところ，1)精度が IFPUG 法の-4%～+11%の範囲に納まっており，2)仕様の明確でない開発の前段階でも要素機能の抽出が容易であり，3)計測速度が IFPUG 法の 3 倍～5 倍速く，開発の前段階での見積技法として，十分なものであることを確認できた．

したがって，目標 1 を達成できたと考えてよい．

(2)については，プロナビ WBS という WBS モデルと，それに基づいたプロジェクト管理システム「プロナビ」を開発し，評価した．

プロナビにより,プロジェクトの工程，作業，成果物，参考資料等の情報を WBS モデルにより相互に関連付けし，一元管理することで，1)プロジェクト計画時の工程，作業，成果物の明確化，2)プロジェクト進捗状況の把握，3)プロジェクトメンバ間での成果物の共有，4)規則，標準，ワークシート等の組織に蓄積された知識の活用，5)開発プロセス及び作業の標準化を実現し，多数のプ

ロジェクトメンバが複数の開発拠点に分散している大規模なエンタープライズ系ソフトウェアの開発プロジェクトでのプロジェクト管理の効率化に図ることができた.

また，プロナビを利用した 100 プロジェクトの管理者に対してのアンケートにより満足状況の評価を行ったところ，109 名から回答があり，95%の人が満足していることが確認できた.

個々の機能に関しては，成果物等の情報の一元管理に最も満足したと回答した人が 34%, 分散した複数の開発場所から利用できることに最も満足したと回答した人が 33%であった. 即ち, プロジェクト計画, 成果物, 組織共通の知識, プロジェクト進捗状況等の情報が，プロナビ WBS により相互に関連付けられて一元管理され，それらをプロジェクトの誰もが参照や利用できることに満足していると回答した人が 2/3 を超えている.

更に，プロナビは平成 12 年の適用開始以来，6 年間で 2000 を超えるプロジェクトで利用されてきており，現在も常時，200 を超えるプロジェクトで使い続けられているという実績からも，プロジェクト管理を支援する有用なシステムであると評価できる.

一方，進捗状況の集計に関する改善要望があったが，これに対しては，(3)で,「プロナビ」の進捗管理機能の強化を図った．この機能強化により，1)進捗情報の集計の機械化により，タイムリーな進捗の把握，2)恣意性を排除でき，成果物の実体に基づいた進捗情報の取得，3)機能単位に分割した成果物の状態を把握することで,どの機能の成果物の作成が遅延しているかの機械的に特定,が可能になった.

したがって，(2),(3)の研究により，目標 2 を達成できたと評価できる.

(4)については，仕様発散防止 QFD という，QFD の定量化，可視化，重点化志向という特徴を取入れた，仕様の早期確定技法を開発した.

この技法を 30 のプロジェクトで適用した結果，約 90%のプロジェクトで，仕様の早期確定に効果があったという回答をしており，目標 3 の達成に貢献できていると評価できる.

これらのことから，本研究では当初の目標を達成したと考えているが，エンタープライズ系ソフトウェア開発に関して，一層，有効な技術，技法の研究に取組んでいきたい.

7.2 今後の研究方針

今後も，エンタープライズ系ソフトウェアの開発を成功に導くための，ソフトウェアエンジニアリングに関する研究を継続したいと考えている．特に，定量的な手法の確立を目指したい．

(1)第 3 章で説明した見積支援システム「AP-Estimate」と，第 4 章，第 5 章で説明したプロジェクト支援システム「プロナビ」を利用することで，エンタープライズ系ソフトウェア開発に関する開発規模，開発工数，開発期間の見積精度の改善を図ることができる．これは第 2 章で説明した COCOMO, COCOMOⅡの拡張になる[8],[10],[12].

AP-Estimate で見積をした結果を基に,プロジェクト計画が作成されプロナビに登録される．プロジェクトの進捗状況はプロナビにより自動収集されるが，進捗は機能毎に FP と成果物数で算出される．したがって，開発規模(FP)，開発工数，開発期間の計画値，実績値をプロジェクト終了までトレースでき，その結果は AP-Estimate の見積 DB と,プロナビのプロジェクト情報 DB に蓄積される．この膨大な情報を分析し指標化することで，開発規模から開発工数，開発期間の算出が，COCOMOⅡよりも容易に，高精度でできることが期待できる．

(2)第 2 章で説明した EASE プロジェクトでの観測型プロジェクト管理支援システム EPM は,プログラムそのものを作成していく過程での定量的なデータを収集している[17],[43]．このデータと，(1)で説明した AP-Estimate やプロナビで収集したデータをあわせて分析することで，プログラム作成の生産性や品質に関する精度の高い指標の設定を期待できる．

参考文献

[1] A. Abran, P. N. Robillard, "Function Point Analysis: An Empirical Study of its Measurement Processes," IEEE Transactions on Software Engineering, vol.SE-22, no.12, pp.895-909, 1996.

[2] A. J. Albrecht, "Measuring Application Development Productivity," Proc. of the Joint SHARE, GUIDE, and IBM Application Development Symposium, pp.83-92, 1979.

[3] A Guide to the Project Management Body of Knowledge 2004edition, Project Management Institute, Inc., Newtown Square, 2004.

[4] 赤尾洋二，品質展開活用の実際，日本規格協会，1988.

[5] 赤尾洋二，品質展開入門，日科技連，東京，1990.

[6] D. Banker, H. Chang, C. Kemerer, "Evidence on economies of scale in software development," Information and Software Technology, vol.35, no.5, pp.275-282, 1994.

[7] C. Behrens, "Measuring the productivity of computer systems development activities with function points," IEEE transaction on Software Engineering, vol.SE-9, no.6, pp.648-652, Nov. 1983.

[8] B. W. Boehm, Software Engineering Economics, Prentice Hall, Englewood Cliffs, N.J., 1981.

[9] B. Boehm, T. Gray, T. Seewaldt, "Prototyping vs. specifying: A multi-project experiment," IEEE Transactions on Software Engineering, vol.SE-10, no.3, pp.290-303, May 1984.

[10] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, R. Selby, "Cost models for future software life cycle processes: COCOMO2.0," Annals of Software Engineering, vol.1, pp.57-94, 1995.

[11] B. W. Boehm, A. Egyed, D. Port, A. Shah, J. Kwan, R. Madachy, "A stakeholder win-win approach to software engineering education," Annals of Software Engineering, vol.6, pp.295-321, 1998.

[12] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer and B. Steece, Software Cost Estimation With COCOMOⅡ, Prentice Hall, New Jersey, 2000.

[13] A. M. Cuelenaere, M. J. van Genuchten, F. J. Heemstra, "Calibrating software cost estimation model: Why and how," Information and Software Technology, vol.29, no.10, pp.558-567, 1987.

[14] T. DeMarco, Controlling Software Projects, Yourdon Press, New York, 1982.

[15] J. Brian Dreger, Function Points Analysis, Prentice-Hall, N. J., 1989.

[16] Department of Defense handbook Work Breakdown Structure(MIL-HDBK-881), Department of Defense, USA, 1998.

[17] EASE プロジェクトにおけるツールと分析手法, EASE プロジェクト　エンピリカル工学ラボ，May 2006.
[18] Q. W. Fleming, J. M. Koppelman, Earned Value Project Management, Project Management Institute, Newtown Square, Inc., 2000.
[19] Function Point Counting Practices Manual, Release 4.0, International Function Point Users Group, 1994.
[20] 福山峻一，高木英雄，田中僚史，渡辺道広，中林效,“ソフトウェアプロセスの持続的な改善を誘導するチェックリストの実装手順,” 情報処理学会論文誌, vol.42, no.3, pp529-541, Mar. 2001.
[21]R. B. Grady, Practical Software Metrics for Project Management and Process Improvement, Prentice-Hall, Englewood Cliffs, N.J., 1992.
[22] Gregory T. Haugan, Effective Work Breakdown Structures, Management Concepts, Vienna, 2002.
[23] Watts S. Humphrey, Managing the Software Process, Addison-Wesley, Winthrop, 1989.
[24] http://www.nesma.nl/
[25] 箱嶋俊哉，“EPM を支える IBM の統合 PM ツール -PMOffice Enterprise,” プロジェクトマネジメント学会誌, vol.6, no.6, pp.33-34, 2004.
[26] "IEEE Standard Glossary of Software Engineering Terminology," IEEE std610.12-1990, 1990.
[27] IT ユーザとベンダのための定量的見積りの勧め，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,(株)オーム社,東京, 2005.
[28] C. Jones, "Measuring Programming Quality and Productivity," IBM Systems Journal, vol17, no1, pp39-63, 1978.
[29] C. Jones, Programming Productivity, McGraw-Hill, New York, 1986.
[30] C. Jones, Assessment and Control of Software Risks, Prentice Hall, New Jersey, Feb. 1994.
[31] C. Jones, Patterns of Software Systems Failure and Success, International Tomson Computer Press, London, 1995.
[32] C. Jones, Applied Software Measurement, The MeGraw-Hill Companies, New York, 1996.
[33] C. Jones, Estimating Software Costs, McGraw-Hill, New York, 1998.
[34] C. F. Kemerer, "An empirical validation of software cost estimation models," Communications of the ACM, vol.30, no.5, pp.416-429, May 1987.
[35] H. Kerzner, Project Management, John Wiley & Sons, Inc., New York, 2001.
[36] H. Kerzner, Strategic planning for management using a project management maturity model, John Wiley & Sons, Inc., New York, 2001.

[37] B. A. Kitchenham, "The problem with function points," IEEE software, vol.14, no.2, pp.29-31, Mar. 1997.

[38] Shinji Kusumoto, Fumikazu Matukawa, Katsuro Inoue, Shigeo Hanabusa, Yuusuke Maegawa, "Estimating effort bu use case points: Method, tool and case study," IEEE Computer Society Proceedings of the 10th International Symposium on Software Metrics, pp.292-299, Sep. 2004.

[39] 柏本隆志，楠本真二，井上克郎，鈴木文音，湯浦克彦，津田道夫，“イベントトレース図に基づく要求仕様書からのファンクションポイント計測手法，”情報処理学会論文誌，vol.41, no.6, pp.1895-1904, Jan. 2000.

[40] 共通フレーム 98 –SLCP-JCF98-(1998 年版),SLCP-JCF98 委員会,（株）通産資料調査会，東京，1998.

[41] 経営者が参画する要求品質の確保，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,（株）オーム社，東京，2005.

[42] G. C. Low, D. Ross Jeffery, "Function points in the estimation and evaluation of the software process," IEEE Transactions on Software Engineering, vol.16, no.1, pp.64-71, Jan. 1990.

[43] 文部科学省リーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」, Annual Report 2005, 奈良先端科学技術大学大学院, 大阪大学, 2006-3.

[44] 西山茂,“ソフトウェア規模の見積り技術の最近の流れ–行数による評価から機能量による評価へ-,” 情報処理学会論文誌，vol.35, no.4, pp.289-298, April 1994.

[45] 野中誠，角瀬章広，ブカーリイサム，東基衞,“画面仕様書に基づく対話型ソフトウェアの複雑度重みつき機能規模の測定技法,” 情報処理学会論文誌，vol.43, no12, pp3993-4004, Dec. 2002.

[46]（株）NEC, コラボレーション型プロジェクト管理システム ProcessDirector, http://www.sw.nec.co.jp/cced/processdirector

[47] Practice Standard for Work Breakdown Structures, Project Management Institute, Newtown Square, 2001.

[48] R. Rada, J. Craparo, "Standardizing software projects," Communications of the ACM, vol.43, no.12, pp.21-25, Dec. 2000.

[49] D. Reifer, Software Management, 4th ed., IEEE Press, Calif., 1993.

[50] C. R. Symons, "Function points analysis-Difficulties and Improvements," IEEE Transactions on Software Engineering, vol.14, no.1, pp.2-11, Jan. 1988.

[51] C. R. Symons, Software Sizing and Estimating, John Wiley & Sons, Chichester, U.K., 1991.

[52] ソフトウェアエンジニアリング基礎知識体系 SWEBOK2004

[53] 高橋光裕他，情報システムの規模見積り手法，電力中央研究所研究報告 (R93016), 1994.

[54] 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，ソフト・リサーチ・センター，東京，1998.
[55] Eric Uyttewaal, Dynamic scheduling with Microsoft Project2002, J. Ross Publishing and International Institute for Learning, Boca Raton, 2003.
[56] C. E. Walston and J. P. Winters, Applying Use Cases, 2nd Edition, Addison Wesley, Winthrop, 2001.
[57] 山本里江子，吉田裕之，“プロジェクト管理ツールとプロセス管理ツールの連携の実現,”情報処理学会第 102 回ソフトウェア工学研究会，vol.1995, no.11, pp.147-152, Jan. 1995.
[58] L. Zells, Managing Software Projects-Selecting and Using PC-Based Project Management Systems, QED Information Sciences, Wellesley, Mass., 1990.