博　士　論　文

高品質ソフトウェア開発のための
実践的プロジェクトマネジメント技術に関する研究

2006 年 1 月

福　島　利　彦

目　次

第1章 序論

1.1 背景

社会にIT（情報技術）が幅広く浸透するにつれ，経済社会システムにおけるITへの依存度はますます高まっている．特に近年では，ITの活用が個人の生活やビジネスの現場の隅々へと浸透している．この背景には，パソコンの性能の急激な上昇，インターネットの普及，携帯電話の爆発的普及などがある．

このような変化に対して，短期間で高品質なソフトウェア製品を開発するようにユーザの要求は強まる一方であり，ソフトウェア開発企業は，ユーザの要求に迅速かつ的確に応えられるよう，ソフトウェア工学の実践を強化し，その一環としてソフトウェア開発プロセスを改善する必要性が出てきている．

しかし，ソフトウェア開発プロセスの成熟度の向上に取り組む企業は増えているが，ソフトウェア開発能力の向上に欠かせないプロジェクトマネジメント技術の定着はできておらず，ソフトウェア開発プロセスへのソフトウェア工学の実践的な活用は遅れている．現実には，ソフトウェア開発プロジェクトの開始時に，当初に掲げたQCD（品質，コスト，納期）の目標を実現することができると自信をもてるプロジェクトは多くないのが実状である．それどころか，納期が近づくと，進捗の遅れと予算のプレッシャーから品質評価プロセスを不十分なままに終わらせてしまい，結局顧客が不満足で終わるプロジェクトも少なくない．

これらの状況から抜け出すには，ソフトウェア開発プロセスの改善により，スケジュールを詳細に計画し，開発プロセスの成果物を定義し，判定基準をもつことは有効である．しかし，そのソフトウェア開発プロセスを有効に機能させるには，実践的なプロジェクトマネジメント技術を向上させることが必要である．

ソフトウェア開発企業およびプロジェクトマネージャは，高品質なソフトウェアを開発しユーザの要求に応えるために，以下の取り組みを実施していく必要がある．

- ソフトウェア工学を実践に活用する．
- グローバルに通用するプロジェクトマネジメント技術をソフトウェア開発プロセスに的確に導入する．
- 自らの組織にとって効率的かつ効果的なソフトウェア開発プロセスを構築する．

1.2 研究目的

これまで，業務範囲の拡大や，大規模かつ複雑化していくソフトウェア製品の開発を手がけるにあたり，品質向上の課題に取り組む必要性が一層強まる中，以下に示すマネジメント技術を導入し，研究を進めてきた．

（１） 失敗プロジェクトを無くするため，プロジェクトのリスクに素早く対応し，そのリスクが発生しないかどうかを追跡するリスク管理技術[1]の研究．

（２） プロジェクト計画や進捗管理といったマネジメント技術の向上を図るため，PMBOK(Project Management Body of Knowledge，プロジェクトマネジメント知識体系)[2]の研究と，プロジェクトを予測・制御するマネジメント技術である EVM(Earned Value Management，アーンド・バリュー・マネジメント)[3],[4]の研究．

（３） ユーザ要求に関する漏れを防止し要求品質を満たすため，ソフトウェア品質特性[5]に基づいたレビューやテストと，ソフトウェア信頼性評価技術[6],[7]を用いた定量的な品質評価尺度の研究．

さらに，これらのプロジェクトマネジメント技術の定着を図り，プロセス改善のレベル向上を図るために，以下に示すソフトウェアプロセスモデルの研究も並行して進めてきた．

（４） プロジェクトマネジメントプロセスの定着を図るための，CMM (Capability Maturity Model，ソフトウェア能力成熟度モデル)[8], [9]に関する仕組みの研究．

（５） ソフトウェア開発プロセスとプロジェクトマネジメントプロセスの標準化のための，SLCP (Software Life Cycle Process) [10]に関する研究と，ソフトウェア開発が成功するためのシナリオである標準ソフトウェアプロセスの研究．

そこで，本論文では，上記のこれまでの研究を基礎にして，実際のソフトウェア開発プロセスの計測によりソフトウェア製品の品質を予測し，ソフトウェア開発プロセスを適切に制御し，作り込まれたソフトウェア製品品質を評価する実践的なプロジェクトマネジメント技術とその体系化について議論する．

マネジメント技術の向上を推進していく仕組みとして体系化された実践的プロジェクトマネジメント技術は，ソフトウェア開発の現場においてマネジメント技術向上のガイドラインとして活用され，ソフトウェア開発企業とプロジェクトマネージャのマネジメント技術の向上を支援することに役立てることができる．

1.3 論文構成

本論文は，本章を含め全５章から構成されている．第２章ではプロジェクトマネジメント技術を，第３章では獲得したマネジメント技術をマネジメントプロセスとしてソフトウェア開発組織へ定着させる方法論を，さらに第４章では定着したマネジメント技術の弱点を定量的評価によって改善する方法について議論する．つまり，プロジェクトマネジメント技術の段階的な発展に沿って章立てをしている．

第２章では，高品質なソフトウェア製品を開発してプロジェクトを成功に導くた

めに有効である実践的なプロジェクトマネジメント技術について述べる．特に，プロジェクトの開始からプロジェクトリスクを定量的に識別し軽減するリスク管理，定量的なデータを用いてプロジェクトの進捗状況を予測制御するEVM，およびテストにおけるソフトウェア製品品質到達レベルを定量的に評価するソフトウェア信頼性評価技術を取り上げる．これらのプロジェクトマネジメント技術の必要性と考え方，ソフトウェアプロジェクトへ導入し適用していく方法，およびその有効性について議論する．

第3章では，ソフトウェアプロセスの改善技術について議論する．ソフトウェアプロセスは，プロジェクトマネジメント技術を組織に定着させるための枠組みとして必要であり，プロジェクトマネジメント技術の向上に合わせて構築し改善していくものである．特に，グローバルなソフトウェアプロセスのモデルであるCMMについて，その構造と特徴を述べ，実際にソフトウェア開発組織に導入する方法，その有効性について示す．さらに，実際に構築し組織に定着させた標準ソフトウェアプロセスを示す．

第4章では，実際のプロジェクトにおける測定データを用いてソフトウェアプロジェクトの定量的評価を行い，プロジェクトマネジメント技術がソフトウェア製品品質や開発コストの改善を図るのに有効であることを示す．この定量的評価は，定着させたマネジメント技術の弱点の改善を持続させるために有効であることも議論する．

第5章では，本論文のまとめを行う．

以上の本論文の構成を図1.1に示す．

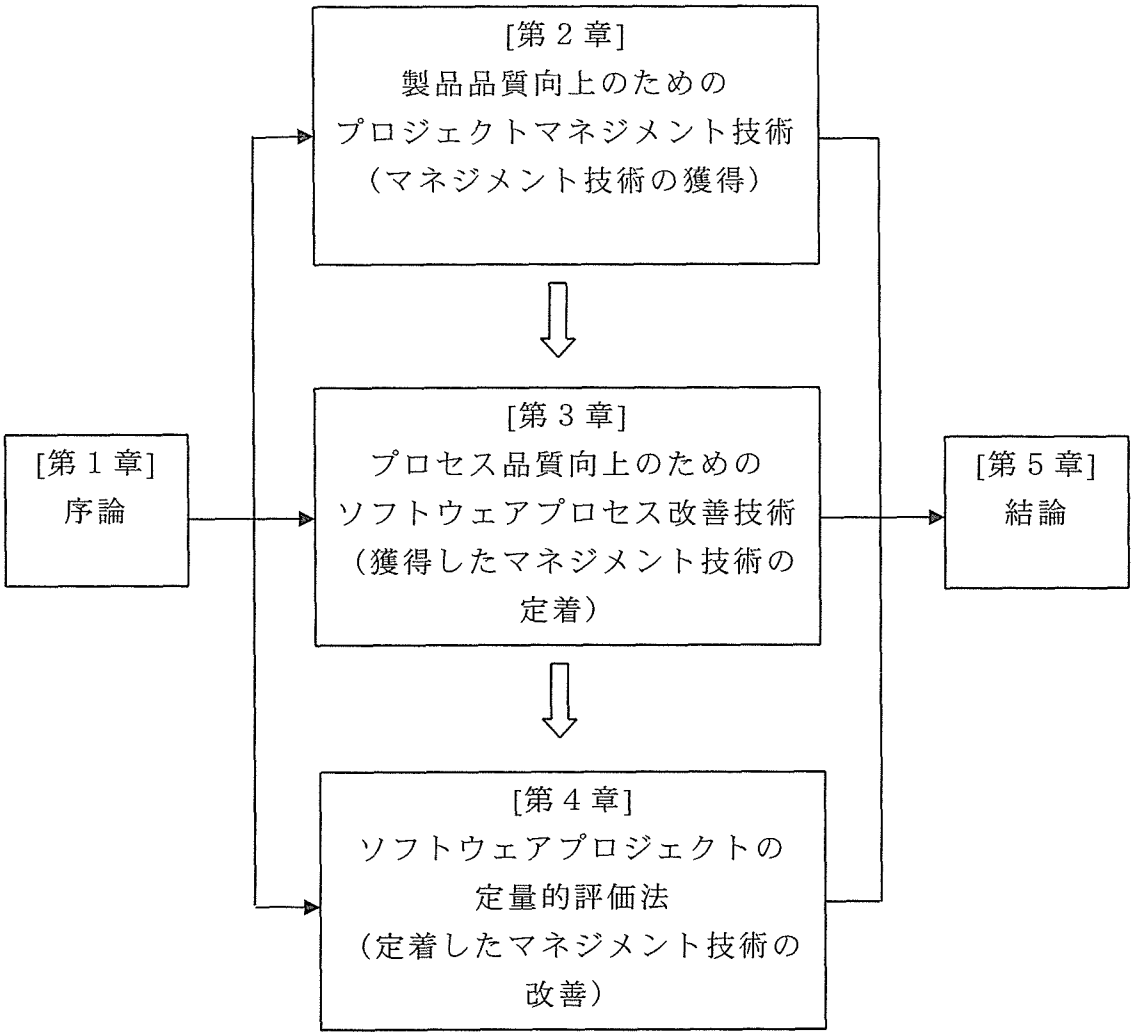

[第 2 章]
製品品質向上のための
プロジェクトマネジメント技術
（マネジメント技術の獲得）
[第 3 章]
プロセス品質向上のための
ソフトウェアプロセス改善技術
（獲得したマネジメント技術の
定着）
[第 4 章]
ソフトウェアプロジェクトの
定量的評価法
（定着したマネジメント技術の
改善）
[第 1 章]
序論
[第 5 章]
結論
⇩は，マネジメント技術の段階的発展を示す

図 1.1 本論文の構成

第２章 製品品質向上のためのプロジェクトマネジメント技術

高品質なソフトウェアを開発してプロジェクトを成功に導くためには，その品質を予測しながら，ソフトウェア開発プロセスに対して適切な制御を実施し，作り込まれたソフトウェア製品品質を評価する定量的なプロジェクトマネジメント技術が有効である．

本章では，プロジェクトの開始からプロジェクトリスクを定量的に識別し軽減するリスク管理，定量的なデータを用いてプロジェクトの進捗状況を予測制御するアーンド・バリュー・マネジメント(EVM)，ソフトウェア製品品質を保証する品質保証技術，およびテストにおけるソフトウェア製品品質到達レベルを定量的に評価する信頼性評価技術について述べ，製品品質向上のためプロジェクトを予測し制御するプロジェクトマネジメント技術とその有効性について議論する．

2.1 マネジメント技術の重要性

ソフトウェア開発プロジェクトが成功するためには，ソフトウェアライフサイクルに基づいて実際にソフトウェアを開発する技術と，ソフトウェア開発プロセスを計測し制御するプロジェクトのマネジメント技術が優れていることが必要である（図 2.1 参照）．

近年のソフトウェア工学では，ソフトウェアの開発技術だけでなく，開発作業を円滑に進めて予定の期間とコストの範囲内で品質の高いソフトウェアを開発していくプロジェクトマネジメント技術が，重要視されている[7]．

ソフトウェアを開発する作業に直接に関係するソフトウェア開発技術とは，次のようなものである．

（１） ソフトウェアの要求定義技術
　　顧客の要求を理解し，正確に漏れなく記述し，完成イメージを定義する技術．
（２） ソフトウェアの設計技術
　　要求仕様を解析し，実現方法をデザインする技術．アーキテクチャ設計技術とデータ設計技術．
（３） プログラミング技術
　　プログラムを書き下ろす作業の標準化技術と効率化技術．
（４） テスト技術

ソフトウェアが要求仕様どおりに正しく機能するか否かを検証し，誤りや欠陥を検出・修正する技術．

一方，プロジェクトマネジメント技術とは，次のようなものである[7], [11], [12], [13]．

（1） 要件管理技術

顧客の要求を引き出して定義した顧客要件とその変更を管理する技術．

（2） 見積り技術

ソフトウェア開発の開発規模，開発工数，開発期間を見積る技術．LOC やファンクションポイント，COCOMO，PERT など．

（3） リスク管理技術

問題として発生する可能性のある事象を探しだし，軽減策を実施し，追跡管理する技術．

（4） プロジェクト計画技術

作業とその実施方法を，開発規模，プロジェクト体制，進捗状況，結果の評価がわかるように定義する技術．

（5） 進捗管理技術

作業の進捗状況を見えるようにし，計画を外しそうな兆候を察知し管理する技術．アーンド・バリュー・マネジメントなど．

（6） 品質保証技術

ソフトウェア製品品質とプロセス品質を保証する技術．

（7） テストおよびレビュー技術

成果物の誤りや欠陥を検出，確認する技術．評価尺度の技術．

（8） 信頼性評価技術

ソフトウェア製品品質を定量的に測定し，信頼性を解析，評価する技術．

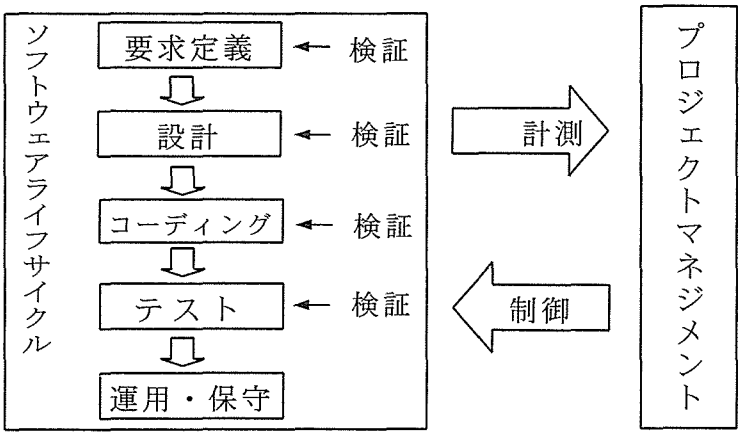


図 2.1 ソフトウェア開発とプロジェクトマネジメント

2.2 リスク管理

リスク管理とはプロジェクトのリスクに立ち向かうマネジメントである．

高品質なソフトウェアを開発し，プロジェクトを成功に導くためには，プロジェクトのリスクに素早く対応し，リスクが発生しないかどうかを追跡するリスク管理技術が有効である．

リスクとは，まだ起きていない問題であり，リスク管理は問題が発生する前の抽象的な概念の段階で対策を考えるプロセスである．災害で例えるなら，30年以内に地震が発生する可能性がリスクであり，地震発生予測や避難所を増やすなどの対策がリスク管理である．地震が発生した後の災害救助は危機管理であり，リスク管理とは言わない．

本節では，リスク管理を実践する方法を議論する．さらに，リスク管理がプロジェクト結果に及ぼす効果を分析し，プロジェクトのリスクに素早く対応しリスクを早期に軽減するリスク管理技術が，プロジェクトを管理するためにきわめて有効であることを示す．

2.2.1 リスクを管理する理由

IT 社会の急速な成長により，ユーザからは短納期でかつ厳密な仕様のシステムを開発するという要求が強まっている．このプレッシャーに対し，プロジェクトマネージャがユーザ要求に適応するマネジメントスキルをもっているとは言いがたい．したがって，ソフトウェア開発には多くのリスクが潜在し，しかも同じようなリスクが繰り返し問題として発生する．問題が発生してから対応するのではマネジメントの失敗につながる．プロジェクトを成功に導くためには，早期にリスクを管理していかなければならない．

これまでの経験プロジェクトでも，過去にユーザ要求が曖昧にもかかわらず，納期と予算が先に決められるというリスクに対して，要求分析の未熟さのために精度の高い計画を立案できず，QCD の失敗を多発させるソフトウェア開発部門があった．図 2.2 は，中でも大きな失敗をした 3 プロジェクトの失敗原因を表す．これらは，プロジェクト計画の失敗，要求分析の失敗，外部委託管理の失敗，進捗管理の失敗によるものであった．

このソフトウェア開発部門の 1 年間の失敗プロジェクトをマネジメント単位に分類すると，予算に対するコスト超過の比率（＝超過コスト／売上高）は，図 2.3 の着手前の失敗コストに示すように，要件管理 2.9 %，プロジェクト計画 4.3 %，進捗管理 1.8 %，外部委託管理 2.6 %，品質保証 0.9 %の状況であった．

そこで，このプロジェクト開始時に入り込む失敗原因に対する対策として，失敗防止に最も効果のあるプロジェクトの開始時に成功シナリオを描く「リスク管理活

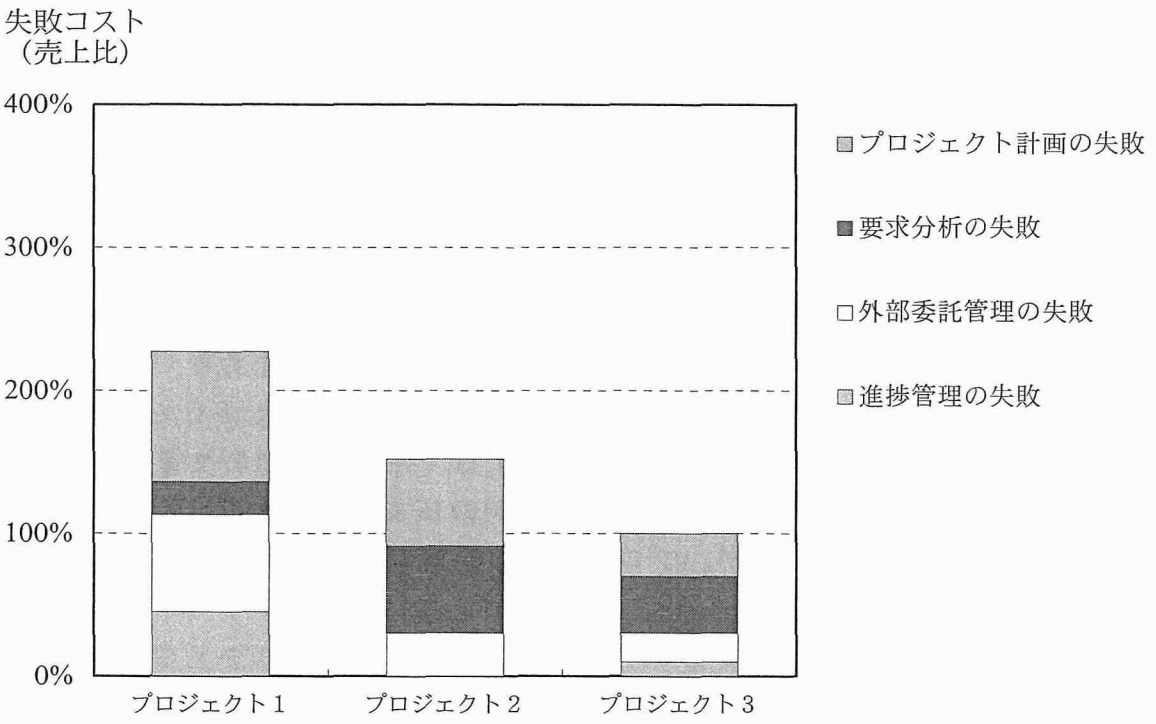


図 2.2　プロジェクトの失敗原因

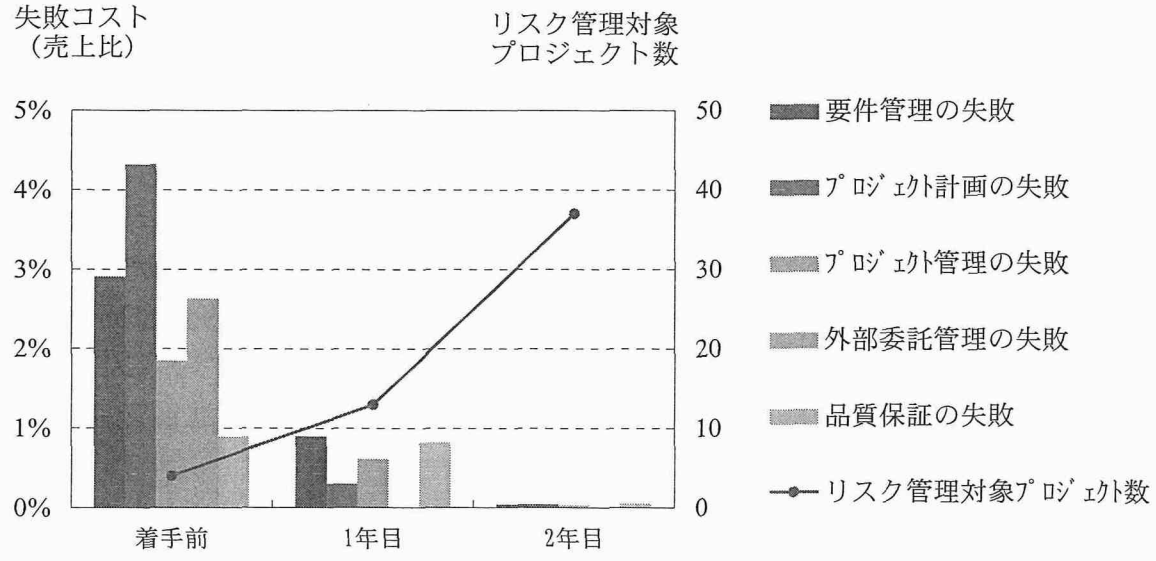


図 2.3 プロセス原因別失敗コストの推移

動」を提案した．なぜならば，「失敗プロジェクトは，いつも計画の甘さやリスクに対する認識の低さに起因する」と分析したからである．失敗プロジェクトの救済のために現場へ入ることで信頼を獲得することによって，リスク管理活動は，失敗を多発させていたソフトウェア開発部門に受け入れられた．

その結果，失敗プロジェクトは大幅に減少し，当初の大失敗撲滅の目標を達成した．このソフトウェア開発部門のコスト超過比率に関しては，12.4 %の減少を達成できた．これは，要件管理やプロジェクト計画が原因の失敗多発の状況が，リスク管理活動により短期間で改善されたことによるところが大きい．すなわち，過去の失敗と因果関係の強いマネジメントの改善は，後戻り作業の減少により，開発組織に生産性向上の効果をもたらすことを証明している．図 2.3 にリスク管理活動によるコスト超過の減少推移を示す．

2.2.2 プロジェクトのリスク

プロジェクトリスクとは，将来起こりうる出来事で，プロジェクトにとって望ましくない結果を生むかもしれないものすべてである．ソフトウェア開発には，多くのリスクが潜在する．あらゆるプロジェクトに現れると予想されるリスクは次の５つと言われる[1].

- 内在的なスケジュールの欠陥
- 要求の増大（要求の変更）
- 人員の離脱
- 仕様の崩壊
- 生産性の低迷

ここで，文献[1]において第１位にあげているのは，スケジュールのリスクである．

PMBOK(Project Management Body of Knowledge) [2]においては，リスクをより具体的に次の４つの区分に分類している．

（１） 技術，品質，性能リスク
　　実証されていない技術，非現実的な性能目標，開発中の適用技術変更など
（２） プロジェクトマネジメントのリスク
　　時間や資源の不適切な割り当て，質の劣る計画，稚拙なマネジメントなど
（３） 組織上のリスク
　　コスト，時間，スコープ目標の不整合，優先順位の欠如，他との資源の競合
（４） 外部リスク
　　制度的環境変化，カントリーリスク，天候異変など

よく検出するプロジェクトのリスクとしては，特にプロジェクトにとって新規性が高い場合，計画が曖昧になり進捗管理が困難になるリスクが多い．その他，見積り時は受注範囲が不明確な要件文書のリスク，計画時は規模見積りの根拠が弱いスケジュールのリスク，開発中はマネジメントスキルの低いプロジェクトマネージャによるリスクなどを日常経験している．

以下に，これまでの経験プロジェクトにおける主なリスクを列挙する．

（１） スケジュールのリスク
- 開発規模の根拠の弱いスケジュール
- 非現実的なスケジュール

（２） 要求定義のリスク
- 新規開発，新技術
- 品質仕様，成果物定義が曖昧
- 受注範囲が不明確
- 運用業務の考察不足

（３） 人的要因のリスク
- プロジェクトマネージャのマネジメントスキル不足
- 要求技術を備えた人材の不足
- プロジェクト内のコミュニケーション不足

2.2.3 リスク管理活動の概要

リスク管理活動の目的は，QCD に関する失敗の可能性をもつプロジェクトリスクの軽減である．本活動は，毎週開催するリスク管理定例会において，プロジェクトマネージャに対しインタビューを実施し，リスクの識別と定量化および軽減策を支援する．開発中はリスクが発生している兆候がないかどうかを追跡する（図 2.4 参照）．リスクを管理する方法は，以下の４ステップを実施する．

（１） リスクの識別
プロジェクトマネージャとのインタビューで，表 2.1 のリスク計算書をチェックシートとして用いて，リスクを分析し識別する．

（２） リスクの定量化
識別したリスクを，リスク計算書によりリスク度として求め定量化する．さらに被害の大きさを推定し，リスク評価を行う．

（３） リスクの軽減策導出

リスク評価結果に基づいてリスク軽減策を計画する．

（4） リスクの追跡と監視

開発中は，リスクを追跡し監視する．追跡の優先順位はリスク評価とリスク度による．

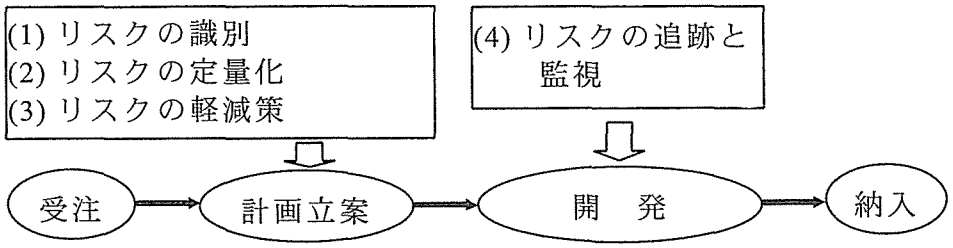


図 2.4 リスク管理活動の概要

2.2.4 リスクの識別と定量化

プロジェクトマネージャに対し，リスク計算書を用いてインタビューを行い，プロジェクトのリスクを分析し識別する．インタビューは，技術部門の開発スキルとマネジメントスキルに精通した専任の SE が実施することによりリスクの識別が可能となる．またリスク計算書は，失敗事例が盛り込まれているので日常のリスク状況に対応しており，インタビューを補助するものである．

リスク計算書におけるリスク項目は，プロジェクトの失敗は再発性が高いことから，過去のプロジェクトからの失敗原因を，システム特性・開発体制特性・顧客特性・契約特性の４つに分類している．識別したリスクにより，式(2.1)によりプロジェクトのリスク度をプロジェクト開始時に計算し，開発中は都度再計算する．

$$\text{リスク度} = \Sigma_i C(i). \qquad (2.1)$$

ここで，C(i)は表 2.1 に示したリスク項目ごとの配点のスコアである．このリスク度は，最大 100 点である．実績では，リスク度 40 点以上のプロジェクトで現実に QCD 失敗を発生させていることが多い．

さらに，リスクが発生したときの被害の大きさを推定し，危険・追跡要・安全の３レベルの重大度評価も行う．重大度の評価結果は，リスク管理活動が取り組むプロジェクトの優先順位となり，危険と判定されたプロジェクトは，リスクの追跡を毎週実施する．

表 2.1　リスク計算書

		リスク項目	配点（スコアC(i)）
システム特性	（1）開発規模	①機能数が多い，又は機能のボリュームが見えない	3
		②システム構造が複雑である．または複雑さが見えない	3
	（2）業務要件	①エンドユーザでのシステムの役割を知らない	4
		②エンドユーザ機能，オペレータ機能があいまい	6
		③ユーザがシステムに要求する品質の重要課題を知らない	4
	（3）性能要件	①性能要件を顧客と合意していない．又は性能を把握していない	4
		②性能要件は，社内での実績がない	4
	（4）拡張性	①ユーザ要件に将来拡張があるのに，最大値を記述した文書がない	2
			30
開発体制特性	（1）技術基盤	①メンバーが経験のない業務，装置に対するシステムである	4
		②スキル不足のメンバーがある	2
		③メンバーが経験のない技術（OS，パッケージ，開発環境，言語）を含む	2
	（2）開発体制	①プロジェクトマネージャの役割と責任範囲が明確でない	2
		②プロジェクトマネージャとしての経験が浅い	4
		③品質保証部門が動作検査すべきであるのに，検査する予算と工期がない	4
		④組織に交代メンバーの余裕がない	2
	（3）スケジュール	①スケジュールは，マネージャが進捗管理できる最少単位に分割されていない	2
		②ソフトウェア成果物の規模（枚数，ステップ数）が見積られていない	2
	（4）資源	①開発環境が不足（ハード発注遅れ，納期遅れなど）	2
		②検証環境が不足（ハード発注遅れ、納期遅れなど）	2
	（5）外部委託	①当社常駐で作業をしない外部委託業者がいる	2
		②外部委託する作業内容（作業範囲と成果物）が明確でない	4
		③開発ソフトウエアの大半が外部委託である	4
		④外部委託先がスキル不足，もしくはその可能性がある	2
			40
顧客特性	（1）工期設定	①見積時に計画した工期が確保できていない	6
	（2）顧客体制	①顧客の仕様決定スキルが不足，又は仕様がころころ変わる	1
		②開発中に他の業務の割り込みを多発する顧客である	1
		③当社で納入実績のない顧客である	2
		④顧客内でコミニュケーションの不足がある	1
		⑤プロジェクト内で，他社もしくは顧客担当分の開発遅れがある	1
			12
契約特性	（1）見積り	①未受注である．又は受注範囲と責任境界が明確でない	4
		②納入成果物名，内容，納入日が決まっていない	4
		③見積りが甘い．または予算が押さえられている	4
		④現地調整が必要だが見積りに正しく含んでいない．又は別見積りにしていない	2
	（2）受入検証	①納品後の顧客受入検証期間と当社の無償保証期間が，決まっていない	4
			18
リスク度			100

2.2.5 リスク軽減策の導出

ソフトウェアプロジェクトは，識別されたリスクの軽減策を計画する．リスク軽減策は，標準ソフトウェアプロセスに基づいてプロジェクトのプロセスを強化し，プロセスの欠陥を防止する．標準ソフトウェアプロセスは，CMM (Capability Maturity Model) [8]に準拠して定義している．実例では，要求分析やプロジェクト計画のマネジメントを強化するリスク軽減策が多い．

新規性が高く何を作るのか要求が曖昧なリスクに対応するリスク軽減策の例を下記に示す．

- 要求仕様書は，顧客の視点で記述され，機能が漏れなく定義され，品質仕様も定義され，検証可能な文書であることを確認するためのレビューを計画する．
- 要求仕様書の顧客との早期合意を計画する．
- テスト項目が要求仕様書を漏れなく抽出していることを確認するためのレビューを計画する．
- ソフトウェア信頼度成長モデル[6],[7]を用いてソフトウェア製品品質到達レベルを評価することを計画する．

2.2.6 リスクの追跡と監視

ソフトウェア開発中は，既存リスクの監視や新たなリスクの抽出のため，リスク計算書を用いてリスク度の再評価を行う．また，リスク軽減策が有効であるかどうかを，進捗管理と成果物レビューの結果により事実に基づいて評価する．EVM を進捗管理の評価に使用する．リスクの影響度が許容範囲内か，問題として発生していないか，といったことを判断するために，リスク管理活動における QCD の基準を下記に示す．

（1） 品質の判断基準
- すべてのレビューに合格しているか？
- 品質保証計画に対しレビュー実施の遅れが 10 %以内か？

（2） コストの判断基準
- コスト予実差が 10 %以内か？

（3） 納期の判断基準
- スケジュール予実差が 10 %以内か？

プロジェクトが，上述の QCD 管理限界を逸脱した場合は，確実にリスクが発生

している兆候である．リスク管理活動は，その原因を分析し，リスク軽減策を立案し，軽減策の有効性を追跡する．

また，管理限界の逸脱が２週間以上継続する場合は，経営層に対してリスク警報を発行し，リスク軽減策を全社レベルで実施する．

2.2.7 リスク管理活動で経験した課題と対策

筆者の行ったリスク管理活動において，最初の２年間で２つの大きな失敗プロジェクトを経験している．その失敗事例を分析した結果，以下の２点を課題として認識し改善した．

（１） 新規性の高い開発プロジェクトのリスク評価

１つ目は，初物（初めての開発）に対する課題である．失敗事例は，仕様書はすでに作成完了し顧客に納入されていた．その後開発も請け負うことになり，その時点でリスク管理活動へ登録されたプロジェクトであった．プロジェクトの開始時には仕様書の掘り下げが浅いと感じたが，顧客と合意済みであったことにより大きなリスクとして取り扱わなかった．しかし，仕様の非現実性が基本設計レビューの度重なる延期によって判明することにより，プロジェクトは，結果的に契約納期に２ヶ月遅れてしまった．以後の再発防止対策としては，初物はハイリスクとして指定したリスク評価を実施している．

（２） より正確なリスク評価の方法

２つ目は，何を作るか曖昧な状態での規模見積りに対する課題である．これもソフトウェア開発ではよくあることである．開始時のリスク評価に見積り金額を使用したことにより，大失敗の可能性を小失敗と誤認してしまった．その金額は，見積もれる状態にない時に見積った精度の低いものであった．失敗事例では，リスク管理活動への登録初期から仕様遅れの問題があり，２ヶ月×２人の遅れを推定し対策案を提案していたが，仕様が明確になった時にはそのソフトウェア規模は計画時の２倍になっていた．以後は，規模見積り精度が低い場合の危険度を再認識し，リスクレビューの中で技術部門に規模見積りの教育を実施している．

リスク管理活動により失敗プロジェクトは減ったとはいえ，２つの失敗事例は活動以前と同じ失敗でもある．これは１つに，品質管理部門が少人数であるという課題が根本原因としてある．同時にフォローしなければならないプロジェクトが２０以上になった頃，プロジェクトへのフォロー活動が浅く広くの傾向になり，結果として２つの大失敗プロジェクトを発生させてしまった．この課題に対する対策とし

て，品質管理部門の活動効率の向上を目指し下記の仕組みを構築し実施した．

ハイリスクプロジェクトが輻輳し始めると危険認識が甘くなる経験から，リスク度評価に加えて，マネジメントプロセス評価ならびにレビュー評価も含めた総合評価によるプロジェクトの優先順位付けに取り組んだ．総合評価では，プロジェクトを“危険”，“追跡要”，“安全”の３つの評価群に分類する．この評価群の中でプロジェクトをリスク度順に並べることにより，リスク管理活動の優先順位が決定する．この仕組みを構築後のリスク管理活動は，リスク管理定例会を開催の都度以下のことを実施し，リスク管理活動のフォロー漏れを無くするようにした．

・リスク計算書によりリスク度を再評価する．
・プロジェクトを総合評価し，プロジェクト管理状況一覧表を作成する．
・プロジェクト管理状況一覧表を用いて，プロジェクトの優先度を管理する．

2.2.8 リスク管理活動の効果分析

リスク管理活動で蓄積したプロセスデータにより，リスク管理活動とプロジェクトの評価結果との相関性について分析する．用いたデータは，2001 年 9 月から 2002 年 12 月の間に，リスク管理活動で危険または追跡と評価された表 2.2 の 18 プロジェクトのデータである．

（１） 開始時リスク度と QCD 管理指標の相関分析

図 2.5～図 2.7 は，プロジェクト開始時のリスク度（横軸）と QCD 管理指標（縦軸）の相関性を分析した．Q（品質）は規模当りの顧客納入後フォールト数であり，C（コスト）は予定コストに対する実コストであり，D（納期）は完了予定日に対する実完了日である．

プロジェクト開始時のリスク度と QCD 管理指標との相関は，最大でコストとの相関係数が 0.4150 と高くない．現実に，要求が固まる前にリスク度評価を実施する場合やリスク評価者のスキルレベルのばらつきにより，プロジェクト開始時のリスク度を定量的に比較することは難しいためである．全体的な相関関係は高くないにもかかわらず，プロジェクト開始時リスク度が 30 点以下の場合，ほとんどのプロジェクトにおいて，顧客納入後フォールト数が 1 以下，コスト超過が 10%以下，納期超過が 10%以下と QCD 目標を達成している．この結果，プロジェクトが安全かどうか評価する管理基準としてリスク度 30 点を使うことができる．さらに開発中には，リスクが安全レベルまで軽減できたかどうかの管理基準として使うことができる．

（２） リスク度軽減速度と QCD 管理指標の相関分析

図 2.8～図 2.10 は，リスク度軽減速度（横軸）と QCD 管理指標（縦軸）との相関

性について分析した．ここで，（リスク度軽減速度）＝（リスク度が 30 点以下に到達するまでの期間）／（開発予定期間）である．開始時からリスク度が 30 点以下の場合，リスク度軽減速度は 0.0，完了時までリスク度が 30 点以上の場合，リスク度軽減速度は 1.0 である．

リスク度軽減速度と QCD 管理指標との相関は，図 2.9 で示されるように，コストとの相関係数が 0.7434 と高い．プロジェクトの早い時期にリスク軽減策に手を打てるプロジェクトは，コスト管理も適正に行えるからである．６プロジェクトが終盤になってもリスク度が下がっていないが，そのうち４プロジェクトに共通する要因は，プロジェクトマネージャとしてのスキルが低い，またはマネジメントが雑であった．全体に，早期にリスクを軽減できるプロジェクトは，QCD 目標を達成する傾向がある．すなわち，リスク度軽減の速さとコスト目標達成との間には高い相関関係があった．

2.2.9 まとめ

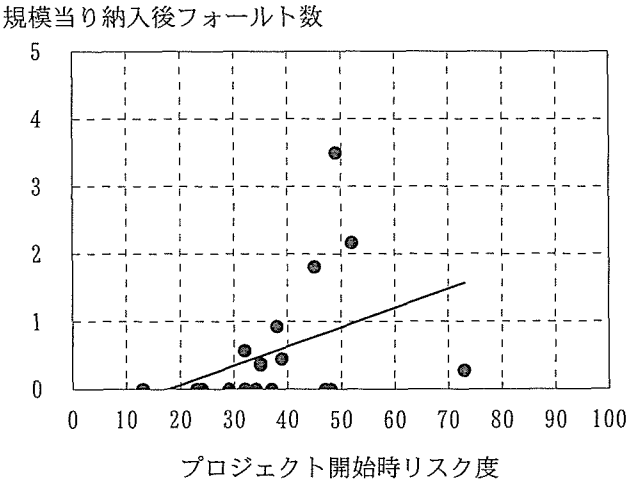
本節では，リスク管理の目的と方法，およびプロジェクト結果との相関について議論した．

結論として，プロジェクト開始時にリスク度が高いほど QCD が未達成の傾向があるが，早期にリスクを軽減すればプロジェクトは QCD 目標を達成できる．QCD に関する失敗撲滅のためには，プロジェクト開始時に入り込むリスクを，プロジェクトの早い時期に軽減するマネジメント技術の実施が有効である．

ソフトウェア開発プロジェクトが QCD で常に成功するために，リスクの評価精度を向上させ，早期にリスクを軽減するリスク管理技術を向上させることは，継続して取り組む目標であるといえる．今後も，リスクを早期に軽減するリスク管理技術と，プロジェクトマネジメント技術の向上に継続して取り組み，プロジェクトのリスクにより積極的により早く対応し，プロジェクトが常に成功することを目指していく．

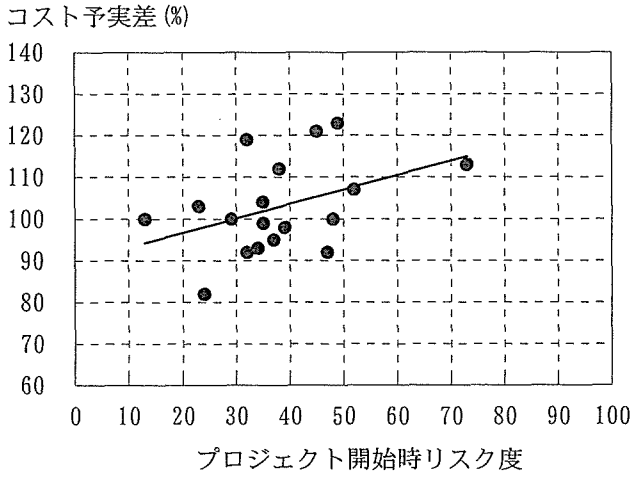
表 2.2　分析データ

プロジェクト No.	開発規模 （人月）	開始時 リスク度 (0～100)	リスク度 軽減速度 (0～1.0)	納入後 フォールト数 (件/規模)	コスト 予実差 (差異無=100%)	納期 予実差 (差異無=100%)
1	3.2	48	0.66	0.00	100%	100%
2	9.9	29	0.00	0.00	100%	122%
3	31.7	35	0.30	0.38	99%	107%
4	9.4	49	0.86	3.52	123%	121%
5	9.0	37	0.17	0.00	95%	96%
6	99.4	45	1.00	1.81	121%	138%
7	8.2	35	0.95	0.36	104%	100%
8	20.9	47	0.33	0.00	92%	91%
9	18.2	34	0.51	0.00	93%	100%
10	42.8	73	1.00	0.28	113%	125%
11	53.9	39	0.33	0.45	98%	100%
12	28.1	32	0.28	0.00	92%	100%
13	16.1	52	0.36	2.17	107%	102%
14	6.5	23	0.00	0.00	103%	100%
15	50.2	24	0.00	0.00	82%	100%
16	1.8	32	1.00	0.57	119%	108%
17	14.0	38	0.87	0.93	112%	100%
18	5.6	13	0.00	0.00	100%	102%



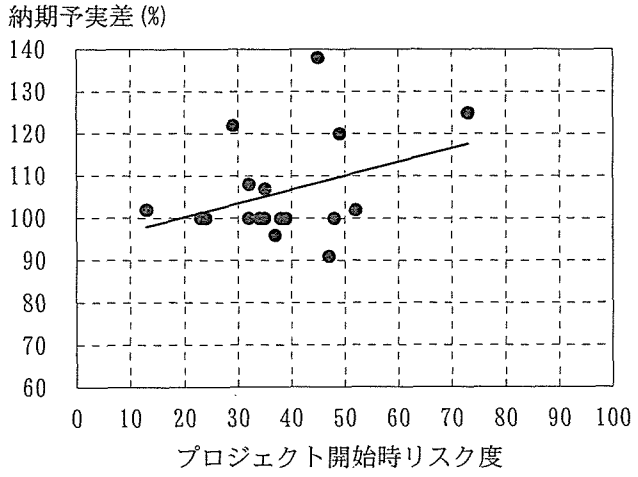
Y=0.0285X-0.5025
R=0.3913

図 2.5 リスク度と品質の相関



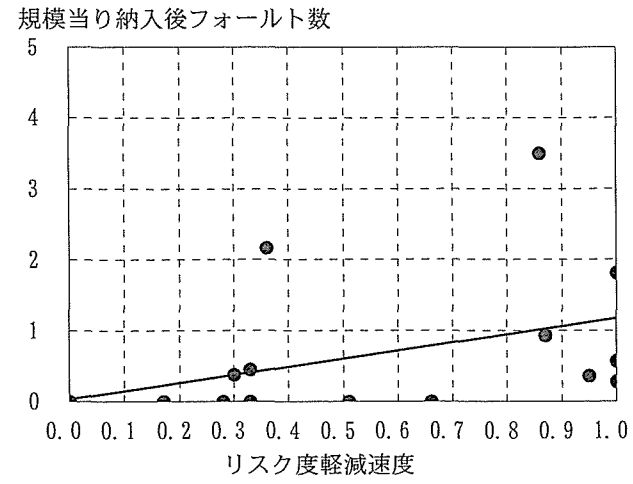
Y=0.0035X+0.8977
R=0.4150

図 2.6 リスク度とコストの相関



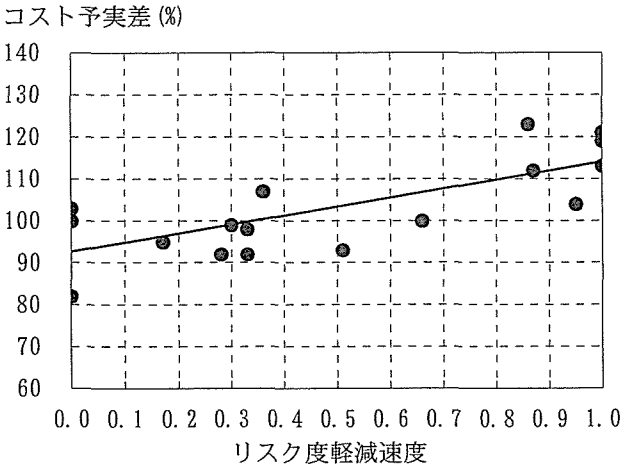
Y=0.0033X+0.9365
R=0.3617

図 2.7 リスク度と納期の相関



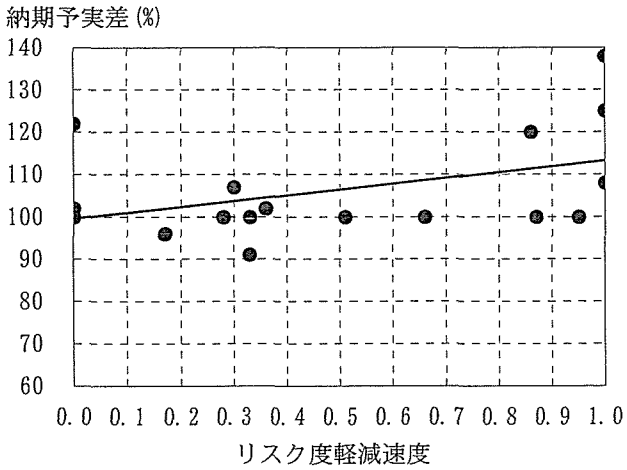
Y=1.139X+0.0351
R=0.4524

図 2.8 リスク軽減と品質の相関



Y=0.2146X+0.9267
R=0.7434

図 2.9 リスク軽減とコストの相関



Y=0.1362X+0.9964
R=0.4330

図 2.10 リスク軽減と納期の相関

2.3 EVM による進捗管理

開発中に計画との差異が生じた場合，その差異の原因を分析して対策をとることが重要である．しかし，プロジェクトにおけるスケジュールのリスクを管理・制御していく上で，プロジェクト計画や進捗管理といったプロジェクトマネージャのマネジメント技術を向上させることは，難しい課題である．この課題に対して，プロジェクトマネージャのマネジメント技術の向上を図り，ソフトウェアプロセスの改善を推進するには，プロジェクトを予測・制御するマネジメント技術である EVM (Earned Value Management，アーンド・バリュー・マネジメント)[3],[4]の導入が有効であると言われている．

我々の経験プロジェクトでは，定量的なデータの測定により，「コストとスケジュールにおけるプロジェクト状況の可視化を実現し，予測性・制御性を向上すること」を目指して EVM を導入した．その結果，プロジェクトの目標値を客観的に予測できるようになった．また，プロジェクトのリスクを早期に軽減する効果があった．本節では，EVM の考え方，分析方法，実践事例，およびその有効性について議論する．

2.3.1 EVM の導入目的

全社的なリスク管理活動と CMM の導入によるプロセス改善活動において，全ての技術部門のプロセス成熟度を向上させていく目標に対し，改善活動の妨げとなるのは，プロジェクト計画，進捗管理などのマネジメント技術が未熟な部門の存在であった．このマネジメント技術が未熟な部門を向上するために EVM を導入した．EVM の導入によるマネジメント技術向上のねらいは，次の 2 点である．

（1） 進捗管理プロセス

進捗管理で重要なのは，計画と照らし合わせることにより，プロジェクトが今後どのように推移していくのか，QCD のゴール地点はどこなのかを予測することである．そのため進捗管理では，開発規模やスケジュールの実績値を測定し，計画値との差を監視する必要がある．

EVM の導入により定量データを測定してそのデータ分析により，コストとスケジュールに関するプロジェクトの可視化を実現できるものと考える．

（2） プロジェクト計画プロセス

これまでの失敗プロジェクトにおいては，改善の PDCA(Plan-Do-Check-Act)サイクルに従って，開発終了時に定性的に失敗原因を追求してきた．EVM の導入により，計画プロセスの能力を定量的に把握することが可能となり，計画プロセスを継続的

に改善するPDCAサイクルが実装できるものと考える.

2.3.2 EVM

PMBOK (Project Management Body of Knowledge)[2]における進捗管理手法として採用されているEVMは，アメリカ国防省の調達規則から発展して，1998年にANSI規格に採用されたものである．コストとスケジュールを関連付けてプロジェクトのパフォーマンスを測定・分析する方法である（図2.11参照）．EVMの分析で必要とされる測定データを下記に示す.

（1） プロジェクト計画時の測定データ

スケジュールから，下記のデータを収集する.

- プロジェクトの開始予定日から完了予定日までの期間： SAC (Schedule at Completion)
- プロジェクトの総予算工数： BAC (Budget at Completion)
- 各作業 (WBS (Work Breakdown Structure)で分解されたワークパッケージ）の開始予定日と終了予定日
- 各作業の計画工数： PV (Planned Value)

（2） 進捗管理時の測定データ

プロジェクトの進捗状況の確認時には，下記の各作業データを収集する.

- 進捗モニタ日
- 現時点までに費やした各作業の実工数： AC (Actual Cost)
- 現時点までに達成した各作業の達成価値： EV (Earned Value)

上記のうち，進捗を管理するために特に重要なデータは， PV, AC, およびEVに関するものである．ここで，EV は，計画工数に対するその時点までに実際に達成した作業価値（出来高）であり，EV＝PV×（達成率）で計算する．例として，PV が100人時のある作業項目対して，現在の進捗が50%達成していれば，EV は50人時である．達成率は最大100 %である．各作業の達成率は，途中経過は概略値で良いが，100 %の達成値を成果物の具体的確認で保証することが重要である．PV, AC, およびEVの各計測単位は，日(Day)でなく時間(Hour)の精度が必要である.

（3） EVMの用語

EVMには，PV, AC, およびEVを含め状態を表す8種類の用語のほか，測定データを分析する5種類の管理指標と，予測に用いる6種類の指標，計19 種類の用語がある[4] ．よく使われる指標を下記に示す．これらの指標により，予算に対する

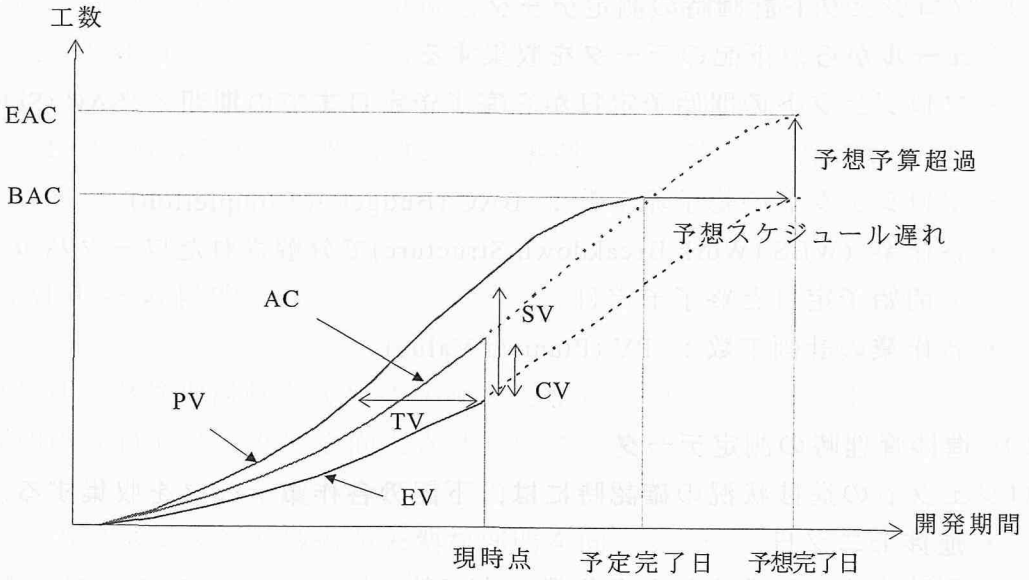

工数
EAC
BAC
予想予算超過
予想スケジュール遅れ
AC
SV
CV
PV
TV
EV
開発期間
現時点
予定完了日
予想完了日

図 2.11　EVM の指標

予実差，スケジュールに対する予実差，および総コストを予測し，プロジェクトが失敗しないかどうかを監視する．

- スケジュール差異： SV (Schedule Variance) ＝ EV－PV（図2.11の縦軸方向計算値）
- コスト差異： CV (Cost Variance) ＝ EV－AC
- 時間差異： TV (Time Variance) ＝ EV－PV（図2.11の横軸方向計算値）
- スケジュール効率： SPI (Schedule Performance Index) ＝ EV ／ PV
- コスト効率： CPI (Cost Performance Index) ＝ EV ／ PV
- 完了時総コスト予測： EAC (Estimate at Completion) ＝ AC ＋ (BAC－EV) ／ (CPI×SPI)

2.3.3 EVMによる分析方法

EVMは，時間単位の詳細なスケジューリングであるとともに，時間遅れが計画の問題かプロジェクト進捗の問題か，遅れの原因を切り分けられるのが特徴である．遅れが生じる場合，もし，コスト管理とスケジュール管理を別々に行っているならば，遅れの原因を定量的に評価することが困難となり，誤って対策を実施するリスクを内在させることになる．

実例を挙げるとEVMでは，ある人の作業項目が5日経過の時点で，達成価値(EV)が24（人・時）の場合は，実工数(AC)により評価は下記の２つに分かれる（図2.12参照）．

（１） パターン１

実工数も 24（人・時）の場合は，実工数と達成価値が重なるグラフとなり，２日分の突発割り込みや担当者の休暇の場合など，２日のみの遅れと評価できる．

（２） パターン２

実工数が 40（人・時）の場合は，実工数と達成価値が異なるグラフとなり，原因は計画精度にある．これは，規模見積りのミス，または担当者のスキル見積りのミスの場合などで，今後もこの比率で遅れが継続する高リスクの可能性がある．遅れの原因を解明しなければならない．

2.3.4 EVMの適用事例

EVMの導入は，進捗管理プロセスにおいて，作業ごとの実工数データ（人・時）の収集を加えることにより実現した．４ヵ月間の試行により，この定量的な計測は

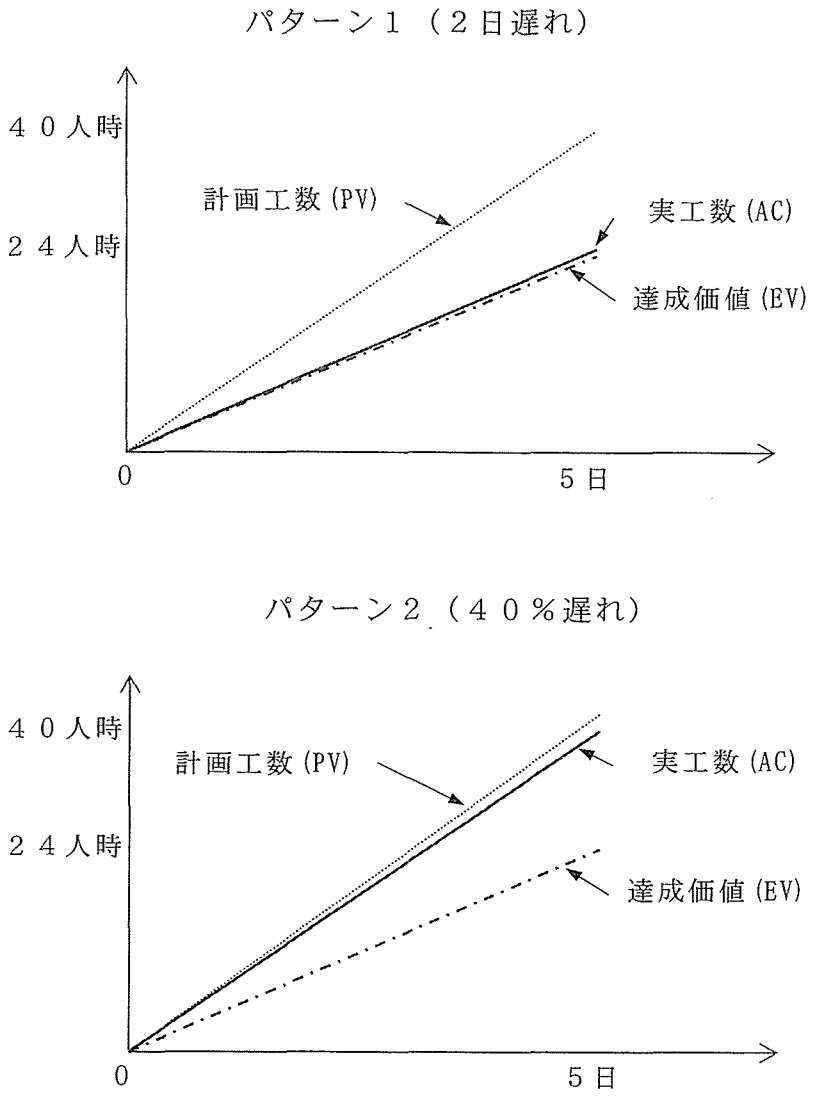


図 2.12 EVM の分析パターン例

プロジェクトの管理精度向上に有効であることを確認した．試行後，プロジェクト計画における工数配分比の管理，進捗管理におけるマンパワー曲線の管理をそれぞれ付加し，本運用を開始した．この２プロセスの改善は次のようなものである．

（１） プロジェクト計画プロセス

プロジェクト計画では，EVM を可能とするため，スケジュールと見積りの一貫性や，全ての作業項目は１項目が 80 時間以下の詳細なワークパッケージまで分解していることを計画レビューで確認する．確認項目は以下の通りである．

- 見積りとスケジュールの整合がとれているか．
- 全ての作業項目はワークパッケージまで分解し詳細であるか．
- 工数の単位は時間で，１項目は 80 時間以下であるか．
- 作業項目ごとに成果物を確認できるか．
- フェーズごとの工数配分比率に問題はないか．
- マンパワー曲線に無理はないか．
- 開発者のスキルを把握しているか．
- 未確定要素があるか．

ここで，フェーズごとの工数配分比率は，COCOMO[11]を参考にプロジェクト標準を設定した．その－10%を管理限界とし，管理限界を超える場合は配分の根拠を確認する．問題がある場合は再計画を要請する．－10%の管理限界は，実際にコスト超過したプロジェクトの原因フェーズが標準配分比の－10%以下であることから決定した（図 2.13 では，要求仕様フェーズの×印，総合テストフェーズの＋印が原因でコスト超過が発生した）．

（２） 進捗管理プロセス

進捗管理では，達成価値(EV)と実工数(AC)を計画工数(PV)と照らし合わせることにより，プロジェクトが今後どのように推移していくのか，QCD の目標達成値はいくらなのかを予測する．コスト差異(CV)またはスケジュール差異(SV) 10 %を管理限界とし，分析による問題の絞り込みと再計画のフォローを実施する．

① 進捗差異の検出

計画と実際の差異を毎週計測し，EVM で危険な状態を見つける．計画に対する偏差－10 %を管理限界とし，管理限界を超える場合は危険な状態と判断する．図 2.14 は，EVM による進捗管理の一例である．達成価値(EV)の×印はスケジュール差異(SV)が管理限界を逸脱した危険な期間であり，期間 31 % (SV= －15 %)で対策を開始している．

② 問題の絞り込み

管理限界を超える場合は，フェーズ単位や機能単位または人単位に EVM

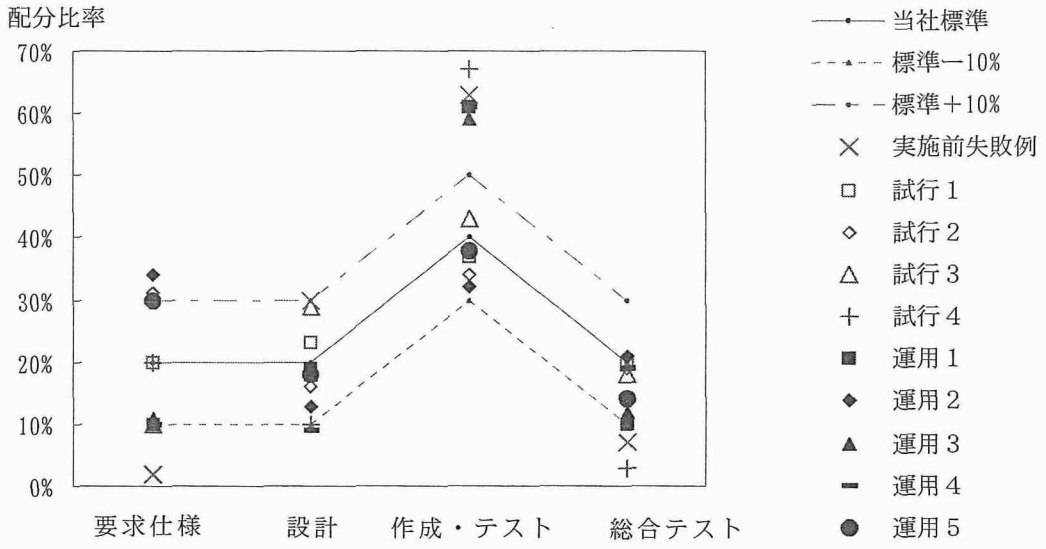


図 2.13 開発フェーズ毎の工数配分比

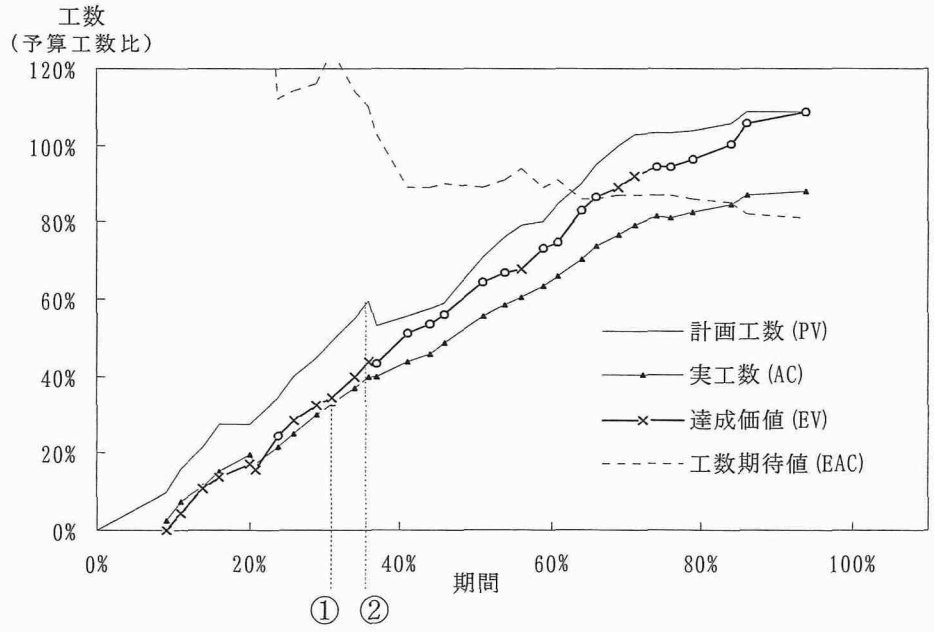


図 2.14 EVM による進捗管理の事例１

を分析し問題範囲を絞り込む．技術部門と共に遅れの原因を解明し，再計画をする．図 2.14 のグラフでは，期間 37 %で再計画を実施し，スケジュール差異(SV)は管理範囲内（○印）に戻っている．

③ 再計画のフォロー

再計画後は遅れが膨らまないよう追跡し，再計画の有効性を評価する．

また，マンパワー曲線を併用して，次の評価も実施する（図 2.15 参照）．

・工程遅延原因は，マンパワーの問題かどうか．

・再計画は，過負荷かどうか．

このようにグラフ化することは，プロジェクト状況の見える管理ができるようになり，プロジェクト管理の初心者に対するマネジメント教育の効果があった．

2.3.5 EVM 導入の効果

EVM 導入のねらいは，進捗管理とプロジェクト計画のプロセス改善であった．その結果，以下の効果を得た．

（1） 進捗管理プロセスの効果

「プロジェクトが今どこにいるのか，ゴールはどこなのか」を客観的にイメージしやすくなった．スケジュールリスクの早期検出が可能となり，QCD 目標に対する予測性・制御性が向上した．

図 2.16 は，EVM 導入後 9 ヶ月までに終了した 8 プロジェクトの納期達成度とコスト達成度の結果である．取り組み目標の指標である計画時の予算±10%，および予定完了日内を逸脱する事例が 2 例発生したが，いずれも管理限界超過にて対策が実施され，小さな失敗の範囲に収まる結果となり，顧客には満足頂けている（●印が管理限界を超過したプロジェクトであり，○印が計画どおり実施できたプロジェクトである）．遅れ対応の早期化に取り組み，全てのプロジェクトを取り組み目標の範囲内に収めることが目標である．

（2） プロジェクト計画プロセスの効果

組織に内在するプロセスの欠陥を，EVM により定量的に抽出し，改善できた．図 2.16 による 2 例の指標の逸脱は，以下のプロジェクト計画プロセスまたはプロジェクトのプロセス定義の問題が深層原因であった．

・事例 1

予算の縮小によるプレッシャーから，計画においてテスト工数を縮小した．その影響で，品質安定化のための検証工程が長期化した問題．

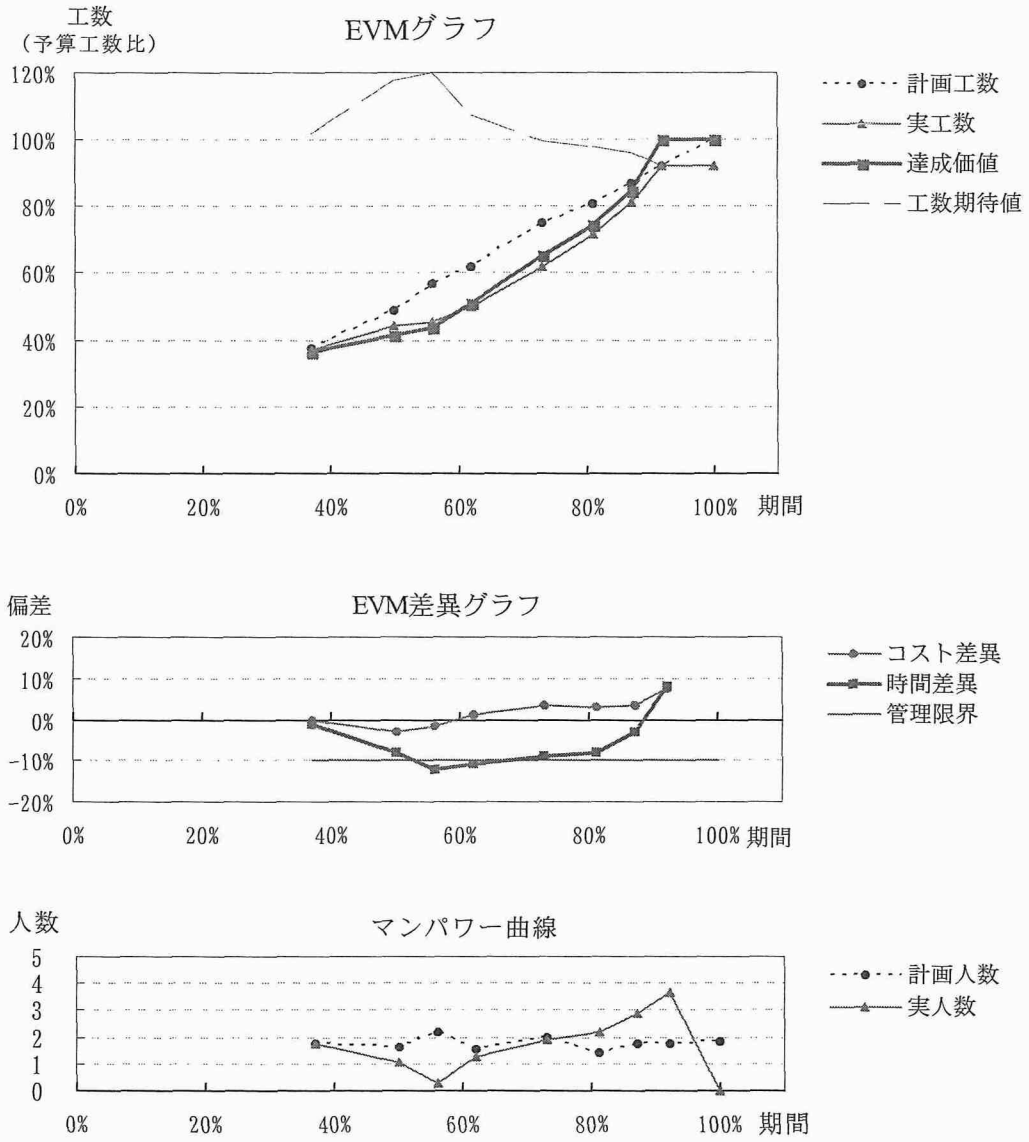


図 2.15　EVM による進捗管理の事例 2

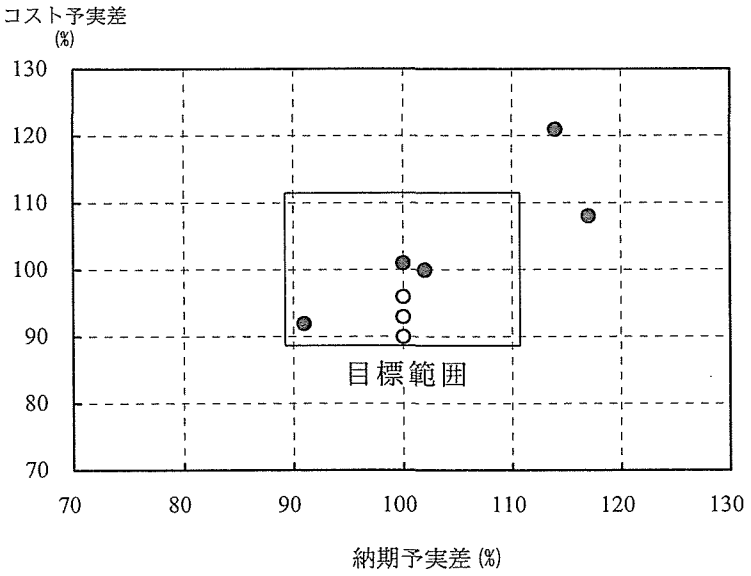


図 2.16　納期目標とコスト目標の達成度

改善策として，プロジェクト計画及び再計画のレビューでフェーズ毎工数配分比をチェックするようにした（図 2.13 参照）.

・事例 2

プロジェクトリーダが，作業範囲を自己解釈して計画していたことにより，開発中に作業工数が増大し工程遅延した問題.

改善策として，要求を正しく捉えることができないで計画する弱点を向上させるため，詳細ソフトウェアプロセスを定義し，ルールとした.

マネジメント技術が未熟な技術部門の場合，なかなか理想的な開発プロセスの実践を実施してくれないものであるが，定量的にプロセスの欠陥を指摘し改善策を指導することにより，プロセス改善を推進させることができる.

2.3.6 WEB システムによる EVM の運用事例

技術部門では複数のプロジェクトが同時進行しているため，マネージャは全てのプロジェクトの進捗状況をモニタし，各プロジェクトの進捗状況が計画と一致していることを常に確認する必要がある．そのために，全プロジェクトの一覧管理が有効である.

我々の経験プロジェクトでは，イントラネットを利用したプロジェクト管理システムを開発し運用した．本システムの主目的は，プロジェクトの進捗状況をモニタし，プロジェクトが QCD に関して成功できるように支援することである．本シス

テムにより,「どのプロジェクトがどのような状態になっているか」を一目で見ることができ，毎日のタイムリーな個別問題に対応できる効果がある．このシステムの主な機能は，以下の通りである．

・プロジェクト計画支援
　スケジュールのフェーズ配分が適切かチェックするため，工数配分比を計算する．
・進捗管理支援
　計測データから，管理指標を計算し，EVM グラフを表示する．さらに，進捗に問題がある場合の原因分析支援のために，個人別の EVM グラフを表示できる．
・人員負荷状況表示
　人員の負荷状況をマンパワー曲線で表示できる．
・品質保証計画支援
　レビュー計画と実績状況をグラフで表示できる．
・プロジェクト状況一覧
　全プロジェクトを危険な順にソートし一覧で表示できる．管理限界超過のプロジェクトは赤色表示される．
・プロジェクト結果の蓄積
　完了したプロジェクトの EVM データを蓄積し，次回以降のプロジェクト計画の支援で活用する．

2.3.7 EVM の効果分析

EVM とプロジェクト結果の相関性を分析した．用いたデータは，2002 年 1 月から 2002 年 12 月の間に,EVM を実施した表 2.3 の 12 プロジェクトのデータである．

（1） EVM による早期リスク軽減効果

図 2.17 は，進捗管理，コスト管理に EVM を適用した 12 プロジェクトの EVM と早期リスク軽減効果の相関関係を表す．早期リスク軽減を測るメトリクスとして，完了予定期間とリスク度 30%以下に達した期間との比率を用いた（2.2.8 参照）．月平均 EVM 回数を横軸にとることにより，EVM の頻度とプロジェクトリスク削減効果の相関性を分析した．

EVM が頻繁なプロジェクトほど，早期にリスクを軽減できる傾向がある，すなわち月平均 EVM 回数とリスク度の軽減の速さとの間には高い相関関係があった．実際，EVM の実施が頻繁なプロジェクトでは，プロジェクトマネージャは，EVM の分析による問題の絞り込みを正しく実行できていた．

（2） EVMによるQCD効果

図2.18～図2.20は,同じ12プロジェクトのEVMとQCD効果との相関性を表す.EVMが頻繁に実施できるプロジェクトほど，QCD目標を達成する傾向がある，すなわち月平均EVM回数とコスト目標達成との間には高い相関関係があった.特に,EVM実施回数が月平均2.8回以上のプロジェクトでは,QCD目標を全て達成していた．毎週EVMの分析を実施できるプロジェクトは，プロジェクトマネージャがリスクを認識し,EVMの分析による問題の絞りこみを行う良好なマネジメントの影響によるものである.

また，QCD目標が未達成のプロジェクトでは，プロジェクトマネージャがEVMの分析による問題の絞りこみから原因を把握できておらず，その結果再計画が頻繁になり，ますますEVMを実施しなくなる傾向がある．この相関分析以降は，粒度を密にした支援でEVMの分析精度を向上させる対策をとっている.

以上の分析結果から，プロジェクトがQCD目標を達成するためには，下記が重要である.

- リスクの評価精度を向上させて，プロジェクトが危険かどうかを早期に見極める.
- 危険と判定したプロジェクトのリスクを早期に軽減するため,EVMを頻繁に実施する.

2.3.8 EVM導入後の課題

EVMは用語の種類が多く複雑に思えるとよく言われるが,実際はごく単純でわかりやすいマネジメント技術である．それでも，EVMを全社に展開していくには，プロジェクトマネージャのマネジメント技術が未熟な場合，有効に活用するのは難しい.「計画と実際の差異の深層原因は何故か？」という課題に，EVMはデータを提供してくれるだけであるからである．考えるのはプロジェクトマネージャである.いかに技術者のマネジメント技術を向上させていくかという課題は,EVMを用いたプロジェクトマネージャへの支援により,継続して取り組んでいくテーマでもある.

2.3.9 まとめ

本節では，EVMの考え方，分析方法，実践事例，およびそのQCD効果について議論した．ソフトウェア開発プロジェクトのリスクを軽減しQCD目標を達成するためには,EVM分析回数の粒度を密に行うマネジメントが有効な管理であることを

示した.

EVM の分析により，プロジェクト計画と実際の進捗の差異を早期に検出，分析，対策し，再計画の精度を向上させることができ，QCD を制御できるようになる．すなわち，EVM は，プロジェクトの目標値を予測するための定量的なメトリクスを得るために有効な技術である.

表 2.3　分析データ

プロジェクト No.	EVM計測回数	EVM回数 (回/月)	リスク度軽減速度 (0～1.0)	納入後フォールト数 (件/規模)	コスト予実差 (差無＝100%)	納期予実差 (差無＝100%)
1	8	1.66	0.95	0.36	104%	100%
2	8	2.29	0.33	0.00	92%	91%
3	19	2.77	0.51	0.00	93%	100%
4	5	2.17	1.00	0.28	113%	125%
5	15	2.54	0.33	0.45	98%	100%
6	7	3.68	0.28	0.00	92%	100%
7	9	2.35	0.36	2.17	107%	102%
8	10	3.09	0.00	0.00	103%	100%
9	26	3.35	0.00	0.00	82%	100%
10	4	2.14	1.00	0.57	119%	108%
11	16	2.74	0.87	0.93	112%	100%
12	9	2.87	0.00	0.00	100%	102%

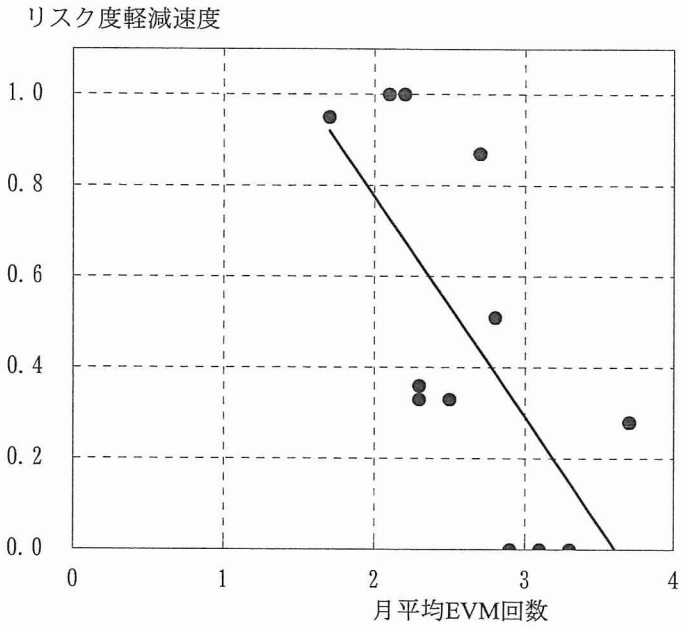
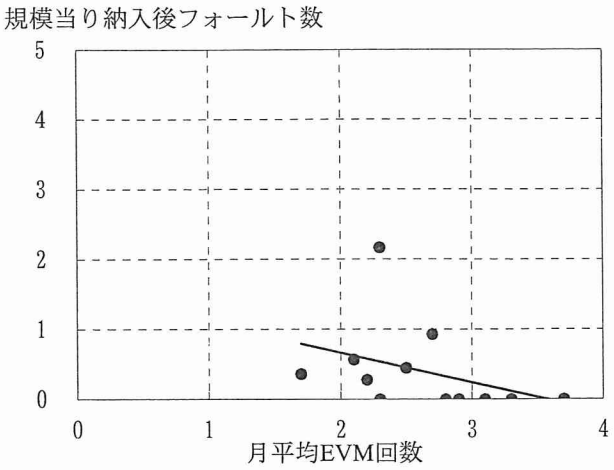
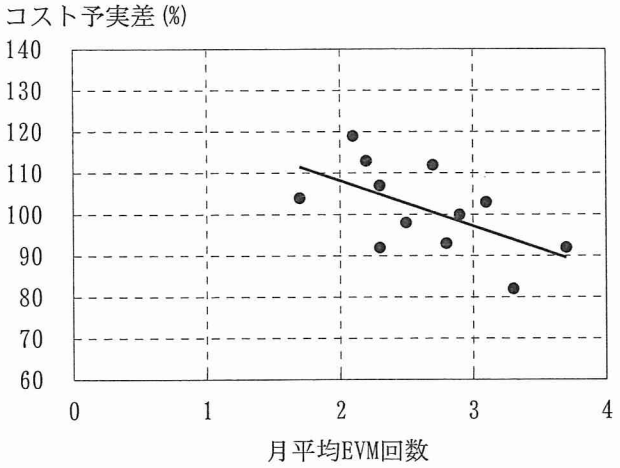


図 2.17　EVM とリスク軽減効果の相関



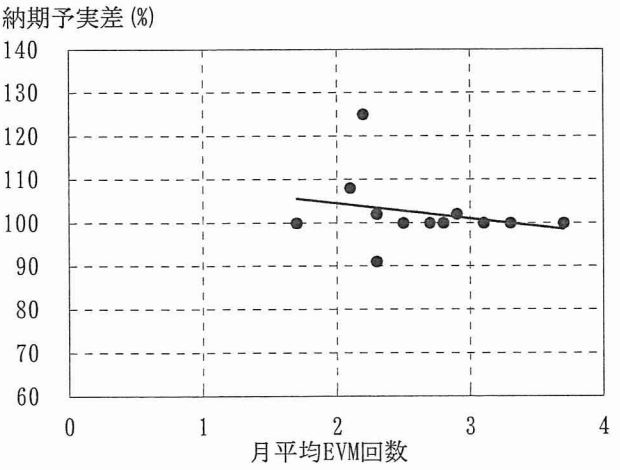
Y=-0.4252X+1.5165
R=0.3776

図 2.18 EVM の Q(品質)効果



Y=-0.1101X+1.3025
R=0.5853

図 2.19 EVM の C(コスト)効果



Y=-0.0351X+1.1157
R=0.2452

図 2.20 EVM の D(納期)効果

2.4 ソフトウェア品質保証

品質保証とは,「顧客にソフトウェア製品の品質を保証します」と言えることが目的である．そのために，下記に示すような，ソフトウェア品質を保証できる仕組みをもっていることが必要である．

- 要求品質を漏らさない仕組みをもっている．
- 要求仕様を実現するための設計であることを確認するプロセスをもっている．
- テスト仕様に要求品質を確実に反映できる仕組みとその確認プロセスをもっている．
- ソフトウェアが完成したことを評価し判定するプロセスをもっている．

本節では，顧客にソフトウェア製品の品質を保証するための，品質保証活動とレビューの進捗管理技術について議論する．

2.4.1 品質保証活動

品質保証活動は，顧客要求どおりのソフトウェア製品を開発し，顧客に納入することを保証するため，成果物の品質をレビューにより確認する活動である．活動項目を下記に示す（図 2.21 参照）．

- プロジェクト計画の立案時に品質保証計画を立案し，ソフトウェア開発部門と合意する．品質保証計画書の抜粋を表 2.4 に示す．技術部門は，レビューフェーズ毎にレビュー予定日，レビュー対象成果物名称，担当者，レビュアを記載する．品質保証部門は，参加するレビューをここで決定する．
- 開発中は，品質保証計画に基づいて要求仕様，設計，テスト項目およびテスト結果のレビューを実施し，成果物の品質が 100%達成されていることを確認する．
- テストでは，ソフトウェア信頼度成長モデル[6]を用いてソフトウェア製品品質到達レベルを定量的に測定し，残存フォールトがほとんどない品質レベルであることを確認する．

2.4.2 レビューの進捗管理

レビューの進捗管理では，計画に対し全てのレビューが漏れなくかつ遅れなく実施されていることを確認する．計画に対する偏差－10%を管理限界とし，管理限界を超える場合は危険な状態と判断し，分析により問題を絞りこむ．図 2.22 は，横軸がプロジェクトの開始から終了までの時間軸であり，縦軸がレビュー実施率およびレビュー評価値である．レビューの進捗は，計画回数累積と実施回数累積の比較に

より状況を確認する．レビュー評価値は，成果物の品質を，設計品質計測シートを用いて定量化したものである．目安としてレビュー評価値が 80%以下の場合は再レビューを実施する．レビュー評価値によって，レビューの進捗が信頼できるようになる．表 2.5 に，システム設計書のレビューで用いる設計品質計測シートの実例を示す．

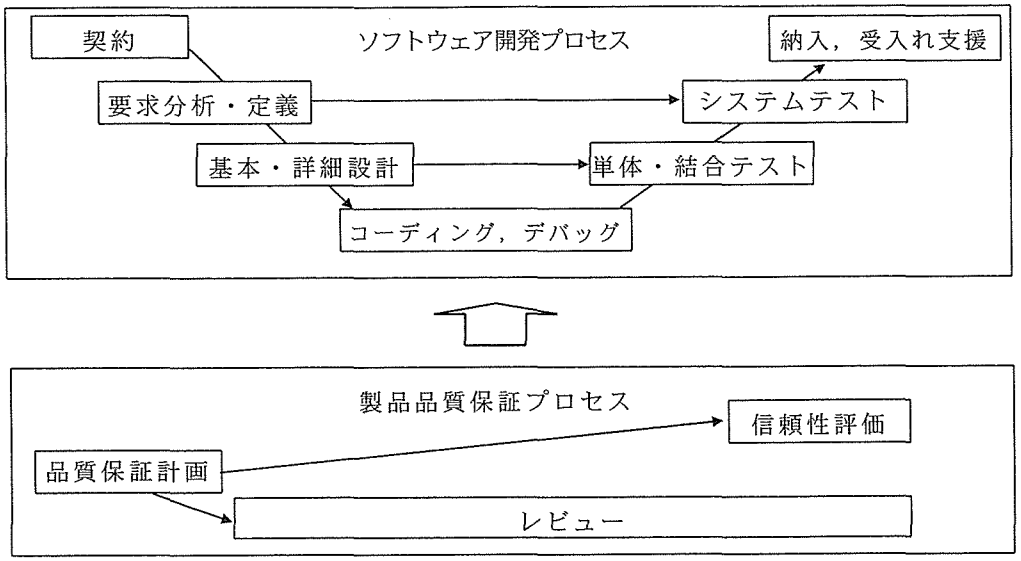


図 2.21 品質保証活動

表 2.4　品質保証計画書の抜粋

レビューフェーズ		技術部門記載範囲						品質保証部門記載範囲				
		予定				実績		予定	実績			
		予定日	成果物名称	担当	ﾚﾋﾞｭｱ	開催日	実績	参加	評価者	評価点	指摘数	修正確認
1	計画											
2	要求仕様											
3	ｼｽﾃﾑ設計											
4	基本設計											
5	詳細設計											
6	ｺｰﾄﾞﾚﾋﾞｭｰ											
7	単体試験											
8	結合試験											
9	ｼｽﾃﾑ試験書											
10	ｼｽﾃﾑ試験結果											
95	品質指標											
96	品質経過報告											
97	検証報告											
98	現地調整											
99	受入検査											

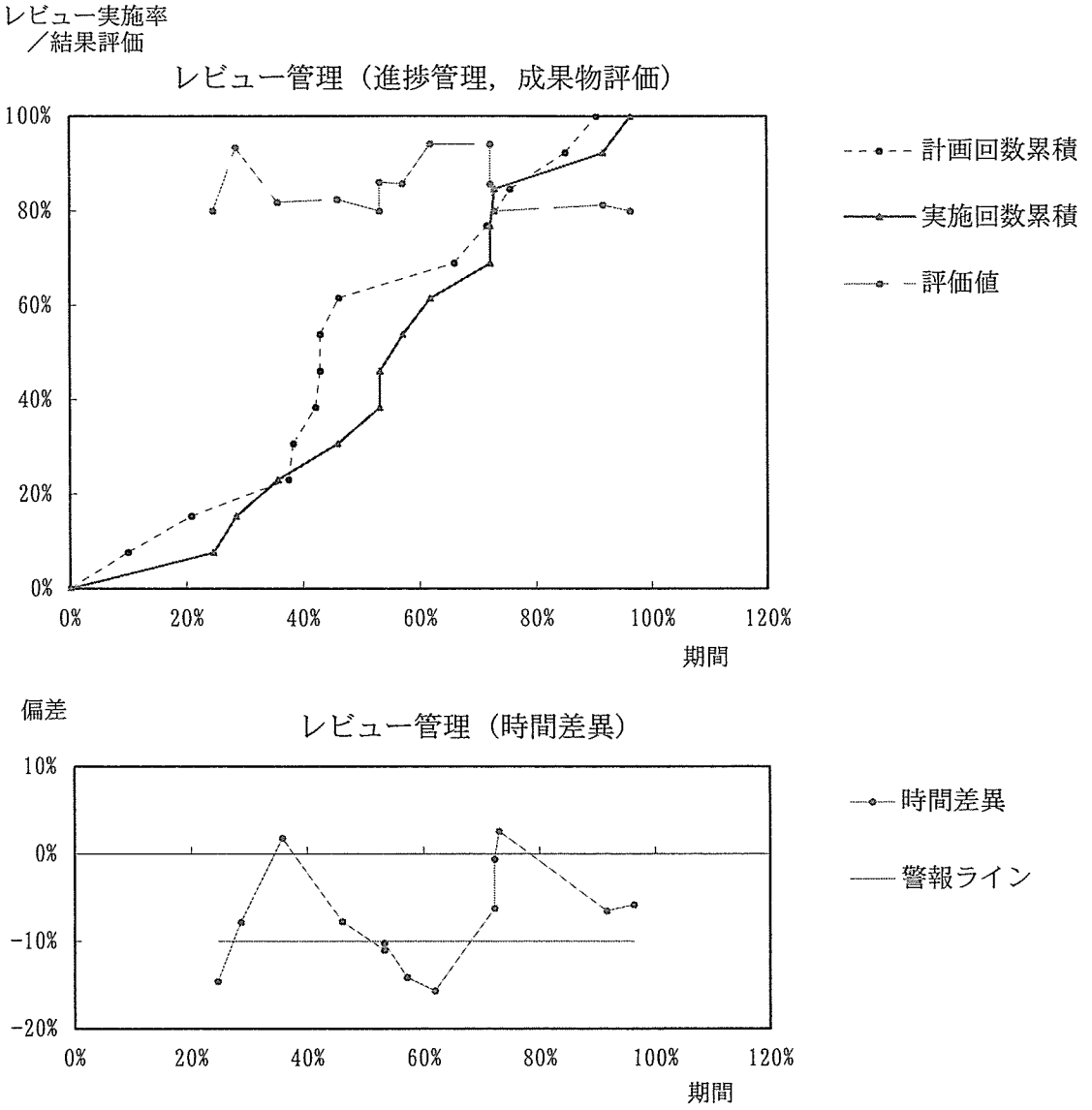


図 2.22　レビュー進捗管理の事例

表 2.5　システム設計計測シートの実例

システム設計	(構成)	①システム全体の装置構成の記載がある ②設計対象範囲の記載がある（受注範囲を明確に） ③各装置説明の記載がある（概要レベルの説明を） ④システム共通で必要な入出力情報を定義している (ファイル，DBやデータ，帳票など)
	(機能設計)	①システムが持つ全機能（もしくは受注範囲内における全機能)の概要説明の記載がある ②上記機能の関連性が明確である ③システムの性能目標値を定義している・通信速度、表示処理速度など ④最大・最小・特異値（=性能限界）を定義している ⑤限界オーバー時の動作・処理を定義している
	(UI設計)	①システム全体で使用する画面のレイアウトや制約，または帳票，ファイルなどを統一している ②画面の概略フローを記述している ③基本操作を行う装置（ex. マウス）と，その操作方法を記述している (起動，終了時の操作方法など) ④ソフトウェアの操作に際し，ユーザ業務の流れに沿っている
ソフト設計		①装置毎に使用されるOSや市販パッケージソフトなどを定義している（バージョン情報も忘れずに） ②上記ソフトの用途を記述している ③仕様追加・変更に対して、容易に対応できるように準備がされている ④他のプログラムに依存しないで自分自身だけで機能を果たすことができる ⑤ソフトウェアのハードウェア依存性を考慮している ⑥ハードウェアの異常検出を考慮している ⑦CPU，通信，メモリ，入出力に関する資源効率を考慮している
異常処理		①システム全体（もしくは責任範囲内）の障害・異常の発生箇所を検知する方法が考慮されている ②ユーザに対して障害・異常の発生をもれなく通知する方法が明記されている ③発生した障害に対して，この障害の影響をくいとめて処理を続行する事ができる ④ロギングなどのメンテナンス用の記録をとる
成果物	(ﾄﾞｷｭﾒﾝﾄ)	①計画・要求された機能とドキュメントに書かれている機能に矛盾がない ②要求仕様書との内容に矛盾がない ③記述項目（言葉）の意味，表現，手順が一義的・同一的である ④設計技法や表記法，用語，記号などが統一されている ⑤ドキュメントに対して，構成，版，それらの整合性や履歴などの製品管理がしやすい
その他		①JIS規格に準じた書き方をしている ②装置毎の仕様と構成について記述している ③動作環境を定義している ④使用する上の制限を記述している

2.5 ソフトウェア品質特性と信頼性評価技術

品質保証プロセスの善し悪しを左右する１つの要因として，テストの品質が関係している．一般的に，ソフトウェア製品に対する品質要求は年々厳しくなり，一義的な基準に当てはめた品質評価だけでは，顧客特有の品質要求事項を満たしていることを評価することは困難である．テストの品質が悪ければ，どんなに時間を費やしても，フォールトを検出しきれず出荷してしまう可能性はある．特に，人的要因に依存するところは品質レベルがばらつく原因でもある．

我々の経験プロジェクトでは，ユーザ要求に対するテストの漏れを防止し品質を向上させるために,テスト品質レベルの向上と品質保証技術の標準化に取り組んだ．本節では，この取り組みにおいて導入した，以下の２つの品質基盤技術について議論する．

（１） ユーザの要求品質に着目したテストの実施によりフォールトを確実に除去するために，ソフトウェア品質特性を定義し，品質指標として標準化されたテスト項目の抽出技術．
（２） ソフトウェア製品品質到達レベルを定量的に評価するための，ソフトウェア信頼性評価技術．

2.5.1 ソフトウェア品質特性

短時間で効率的なテストを行うには，適切なテスト項目の抽出，適切なテストデータの用意などが重要になってくる．これらを実現するためのツールとして,“品質指標”を開発した．品質指標は，ソフトウェア品質特性の側面からテスト項目の抽出をトップダウンに行い,テスト項目の漏れを無くするためのものである.以下に，品質指標を定義する上で導入したソフトウェア品質特性について説明する．

（１） ソフトウェア品質特性とは

ソフトウェアは目に見えない製品であるため，その評価は非常に難しい課題の１つである．反面見えないからこそ，ソフトウェア品質の評価は重要となる．ソフトウェア品質を管理し，制御し，評価する時には，管理対象を知ることが必要になる．その管理対象を知るためのツールとして，ISO9126 で定義されているソフトウェア品質特性[5]がある．品質特性には，それらをさらに詳細化した品質副特性が定められており，全部で品質特性が 6 種類，品質副特性が 21 種類存在する（表 2.6 参照）．

ソフトウェア品質特性に基づいてソフトウェアに対する要求品質を定義する場合は，“みなす”という方法で定義することが重要である．例えば，使用性の要求品質では,「すべての画面は，マウスボタンクリックが２回以内で表示できること」と具

表 2.6　ソフトウェア品質特性

品質特性	説　　明	副特性
機　能　性 (functionality)	ソフトウェアがある目的をもって求められる，必要な機能を実装している度合い	・合目的性 ・正確性 ・相互運用性 ・標準整合性 ・セキュリティ
信　頼　性 (reliability)	実装している機能が，明示された条件の下で機能要件を満たして，明示された期間，正常動作し続けることができる度合い	・成熟性 ・障害許容性 ・回復性
使　用　性 (usability)	ソフトウェアシステムの「使いやすさ」,「使用するために必要な労力」,「使用結果」の良し悪しの指標	・理解性 ・習得性 ・運用性
効　率　性 (efficiency)	明示された条件における，ソフトウェアがもつ目的達成の度合いと，使用する資源の量の関係	・時間効率性 ・資源効率性
保　守　性 (maintainability)	仕様化された改訂を行うために必要な労力に関係する度合い	・解析性 ・変更性 ・安定性 ・試験性
移　植　性 (portability)	ソフトウェアをある環境から他の環境へ移した場合のソフトウェアの能力を推し量る指標	・環境適応性 ・設置性 ・規格適合性 ・置換性

体的に定義すれば，使用性の要求品質とみなせる．またテストも可能となる．このように，ソフトウェア開発工程の入口と出口にあたる要求仕様書とシステムテストでは，ソフトウェア品質特性を用いて具体的にみなすことが重要である．抽象的に「使用性がいいこと」と定義しても，設計とテストはできない．

（2） 導入事例と効果

標準的に定義されているソフトウェア品質特性をそのまま採用することは，品質保証部門においても抽象的であり使えないものである．そこで，ソフトウェア品質特性から実際のソフトウェア製品に即して，技術部門でも容易に使用可能な品質指標へブレイクダウンし，テスト項目抽出の標準化を図った（図 2.23 参照）．表 2.7 はプロジェクトの品質指標をレビューする際に用いるチェックシートである．この品質指標には，独自に安全性特性を付加している．これは，PL 問題の可能性は念入りにチェックしようとの考えからである．

品質指標を導入する以前は，機能面や操作面といった目に付きやすいテスト項目に偏りがちであったものを，導入後は信頼性などの見落しそうな品質特性にも注力することができた．さらに，顧客が特に重要視する特性についてより詳細なテスト項目の設定ができるようになった．導入の結果，テスト項目が充実されたことによりテストでのフォールトの取りこぼしも減少した．つまり，品質指標を基にしたテスト項目の設計により，ユーザ要求の取りこぼしを防ぎ，顧客満足を得られるようになった．

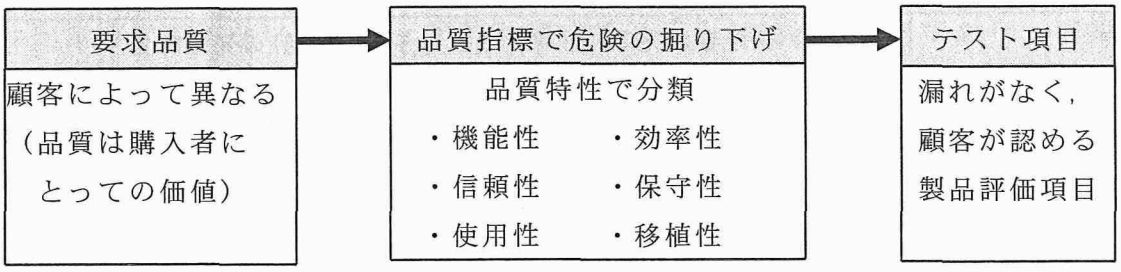


図 2.23　品質指標を用いたテスト項目の作成

2.5.2 ソフトウェア信頼性評価技術

（1） ソフトウェア信頼性評価技術とは

ソフトウェア開発では，品質が安定している確証のもとに，顧客にソフトウェア製品を納入しなければならない．このため，ソフトウェア開発プロジェクトでは，開発工程の管理データを収集して，製品の品質を計測する必要がある．

ソフトウェア製品品質の計測方法としては，ソフトウェア信頼性評価技術がある．

表 2.7 品質指標の測定項目

機能性	(通常機能) (限界値定義) (異常ケースの処理)	①基本仕様書通りの動作をしているか ②仕様書に記述されている数値範囲に対する動作を行えるか (ex. データ保存日数，設定データの入力など) ③実運用に沿った受け付け業務にて問題なく動作するか ④データの特異ケースに関して問題なく動作するか ⑤通信中の操作に関して問題なく動作するか ⑥システムの特異運用 ⑦電源スイッチ操作誤り時 ⑧ＤＢの同時操作にて，データに矛盾は無いか ⑨その他の異常ケース
性能	(特定機能の実行時間)	①応答時間は基本仕様書通りか(通常時・処理の負荷集中時) ②画面表示時間は基本仕様書通りか(通常時・処理の負荷集中時) ③要求された性能を満足しているか ④その他(オフライン作業など)の処理時間
信頼性	(故障を起こさない拡大被害を抑える)	①ハードウェア・ソフトウェア障害時の検出，報告，再試行，自動回復，縮退運転，復旧ができるか ②フールプルーフ／フェイルセーフが考慮されているか ③実運用で安定して動作するか ④メモリーリークに対する対策・検証 (Windows システムでは必須) ⑤過去の不具合事例の対策(水平展開)
安全		①PL 問題(使用者の身体・財産に損害を与える問題)の発生防止 ②セキュリティ対策
操作性	(使用法の習得や操作の容易さ)	①機能の重要度・使用度・使い勝手を反映した画面構成になっているか (ボタン配置，メッセージ，データ表示など) ②重要度の判別が容易に行えるか(重要な操作での再確認など) ③OS，市販ソフトウエアの操作Ｉ／Ｆに準拠しているか(適切な操作ボタンの配置・画面構成，操作性の統一，冗長な操作がないなど) ④使用者が誤操作をした場合のエラーメッセージが表示されるか ⑤マニュアルに従って，設定変更，簡単なメンテなどの対応ができるか
保守性		①使用者による故障発生時の一時切り分け判断ができるか(パソコン，プリンタなどの機器単位での判断可能か) ②システム管理が容易に行えるか(ユーザ管理，データバックアップ，バージョンアップなど) ③拡張が容易に行えるか (基本仕様書に定義してあるハードウェア，ソフトウェアの拡張が容易に行えるか)

その中でも，ソフトウェア信頼度成長モデル[6], [7] は，開発中のソフトウェアに含まれるフォールト数やソフトウェア信頼度を推定する方法としてよく知られている．ソフトウェア品質保証能力の向上のために，ソフトウェア信頼性評価ツール SRET Ⅱ[6]を適用し，実際の開発プロジェクトの出荷品質の判定に適用した．

ソフトウェア信頼性の計測・評価においては，テスト時間や運用時間の計測単位として，カレンダー時間，CPU 時間（実行時間），工数やテスト労力などの連続時間がよく採用される．これに基づいて，ソフトウェアの実行過程，いわゆる動的環境におけるソフトウェアの挙動を信頼度成長過程として記述するものがソフトウェア信頼度成長モデル(Software reliability growth model)である．ソフトウェア信頼度成長モデルは，テストにより発見されたフォールトはすべて修正・除去され，修正時に新しいフォールトは作り込まれないという完全デバッグ(perfect debugging)の仮定の下に，ソフトウェア故障の発生する確率も減少してソフトウェア信頼度が増加する過程を記述するモデルである．

代表的なソフトウェア信頼度成長モデルとして，実施されたテスト時間内に発見されるフォールト数や発生するソフトウェア故障数を観測して，確率則を導入し，ポアソン過程を仮定する NHPP（nonhomogeneous Poisson process，非同次ポアソン過程）モデルと，フォールト発見数データに直接傾向曲線を当てはめて統計解析を行う統計的データ解析モデルがある．従来からよく現場でも使用される NHPP モデルと統計的データ解析モデルを採用した．

ソフトウェアの信頼性評価を行うにあたって導入した支援ツール SRETⅡには，表 2.8 に示す４つのモデルと，テスト労力依存型ソフトウェア信頼度成長モデルの合計５つの代表的なソフトウェア信頼度成長モデルが組み込まれており，信頼性評価を容易に行うことができる．ここで，表 2.8 における平均値関数は，テスト時間区間$(0,t]$において発見される総フォールト数の期待値を表す．また，フォールト発見率とは，残存フォールト１個当りのフォールト発見率を意味する．この指標は，任意のテスト時刻におけるソフトウェア内に残存するフォールトに対する発見難易度を表す．

（２） ソフトウェア信頼度成長モデルの適用事例

ソフトウェア信頼性評価を適用する開発プロジェクトに対しては，プロジェクト開始時に，テスト方針や評価の必要性や結果の見方を説明した．また，テスト開始前には，評価するのに必要なデータをテスト実施担当者と打合わせし，テスト開始と同時に定期的にデータ収集，およびツールによる評価を行った．また，テスト過程の妥当性もチェックし，プロジェクトマネージャへの評価結果報告を行った．

対象システムのテストデータに，指数形ソフトウェア信頼度成長モデルと，遅延 S 字形ソフトウェア信頼度成長モデルの２つのモデルを適用した結果を，表 2.9 に

表 2.8　ソフトウェア信頼度成長モデル

モデル名	平均値関数	パラメータ
指数形 SRGM	$m(t)=a(1-\exp[-bt])$ $(a>0, b>0)$	a：テスト開始前に潜在する総期待フォールト数 b：フォールト発見率
遅延 S 字形 SRGM	$M(t)=a[1-(1+bt)\exp[-bt]]$ $(a>0, b>0)$	a：テスト開始前に潜在する総期待フォールト数 b：フォールト発見率
ﾛｼﾞｽﾃｨｯｸ曲線モデル	$L(t)=K/(1+m\cdot\exp[-ct])$ $(K>0, c>0, m>0)$	K：テスト開始前に潜在する総期待フォールト数 c, m：定数パラメータ
ゴンペルツ曲線モデル	$G(t)=K\cdot a^{(b^t)}$ $(K>0, 0<a,b<1)$	K：テスト開始前に潜在する総期待フォールト数 a, b：定数パラメータ

（SRGM(Software Reliability Growth Model)：　ソフトウェア信頼度成長モデル）

表 2.9　適合性評価結果

適用モデル	適合性評価
指数形 SRGM	K-S 検定により有意水準 5％で適合 （実測値と推定値の偏差 2 乗和：392）
遅延 S 字形 SRGM	K-S 検定により有意水準 1％で適合 （実測値と推定値の偏差 2 乗和：3006）

示す．データの計測時間は，テストケース消化時間である．

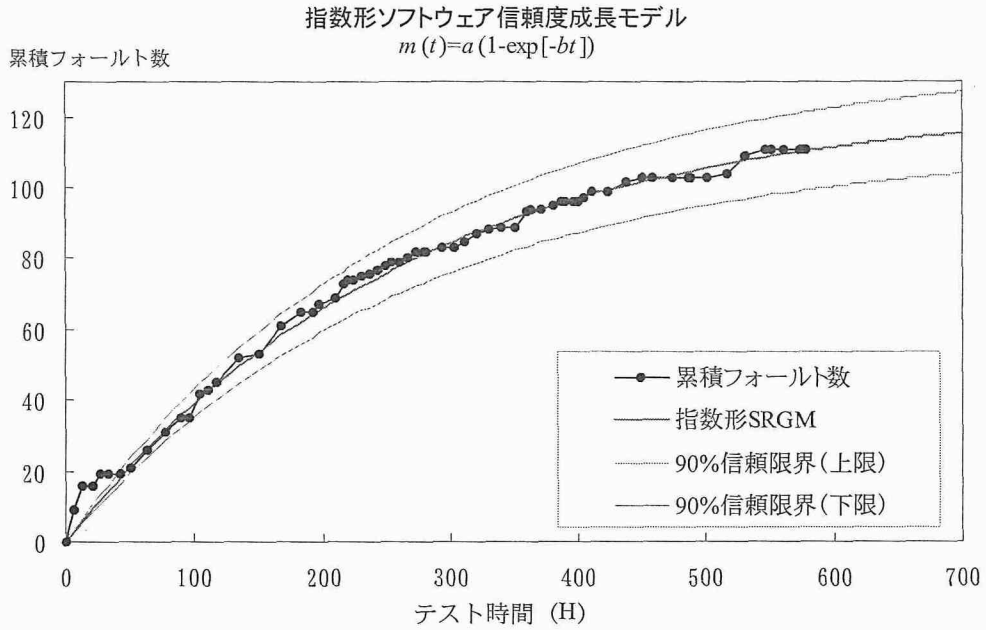
この適合性評価結果より，採用したデータに対しては遅延S字形ソフトウェア信頼度成長モデルの実測値に対する適合性よりも，実測値と推定値との誤差が小さい指数形ソフトウェア信頼度成長モデルの適合性が良いことがわかる．指数形ソフトウェア信頼度成長モデルは，テスト時間を通じて残存フォールト数1個当りのフォールト発見率または残存フォールト1個当りのソフトウェア故障発生率は一様である現象を記述するものである．このシステムは開発規模が大きく，テスト時間の経過とともに発見されるフォールト数を表す成長曲線が指数形を示している.よって，指数形ソフトウェア信頼度成長モデルを適用して信頼性評価を行うのが適当と考えられる．

図2.24は，得られた解析結果の1つである．この事例のプロジェクトは，300時間までに品質指標を用いた全テスト項目を消化し，500 時間まで検出したフォールト修正後の再テストを実施している．500 時間以降のグラフの立ち上がりは，遅れていたハードウェアを組み込んだテストをここで実施したためである．グラフの黒丸(●)は実測データ，中央の曲線が推定曲線であり，上下の曲線は推定曲線の90 %信頼限界を表す．テスト終了時点（579時間）までに発見されたフォールト数は111個あるので，テストにより未発見となったソフトウェア内の残存フォールト数は約14(= 124.2 - 111)個と推定される．テスト終了時にシステム全体の約89 %のフォールトが発見されている．図2.24より，検証開始時はフォールト検出率が高いが，検証工程が進むにつれてフォールト検出率が低くなっている．つまり，ソフトウェアの信頼性が向上しソフトウェア製品品質が安定していくのが分かる．

一般的に，このソフトウェア内の残存フォールト数でソフトウェアの信頼性が評価されることが多いが，その他の信頼性メトリクスであるソフトウェア信頼度や瞬間MTBFでも同様に評価できる．そこで，両者によって検証終了段階における信頼性を評価した結果を以下に述べる．

図2.25に，任意のテスト時点での不良の発見されない確率の時間的挙動を表すソフトウェア信頼度の出力結果を示す．検査579時間経過した時点で製品検査の環境と同じ厳しさで,さらに検査を10時間実施した時のソフトウェア信頼度の推定値は約 0.61 であると推測される．すなわち，約 40%の確からしさで次の不良が検出されると推測される．

また，テスト終了段階における瞬間MTBFを評価した結果が図2.26である．瞬間MTBFは，任意のテスト時刻における平均フォールト発見時間間隔の推定値を示す．簡単に言えば，システムや機械が故障するまでの時間の平均値のことである．グラフよりテスト時刻が進むにつれてフォールト発生時間間隔が伸びていくことが分かることから，確実にフォールト摘出作業が行われソフトウェア信頼度が着実に達成



総期待フォールト数　124.2　　偏差二乗和　392.1857
期待残存フォールト数　13.2　　K-S 検定量　0.0583

図 2.24　指数形ソフトウェア信頼度成長モデルの適用事例

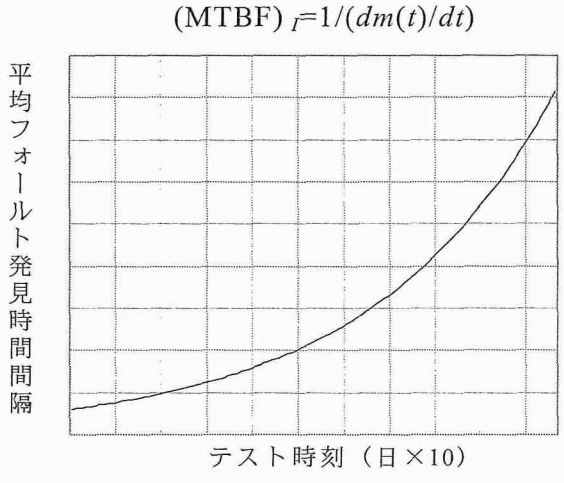


図 2.25　ソフトウェア信頼度の出力結果

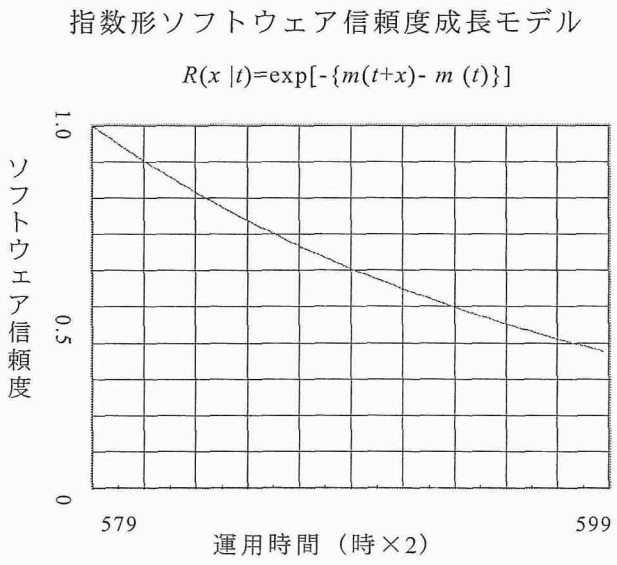


図 2.26　瞬間 MTBF の出力結果

されていったことが読み取れる.

以上の結果を用いて，製品品質の安定度評価を行い，製品の出荷判断を実施している．上記事例の場合，客先への納品後は，開発システムに対するクレーム報告は0件であった.

（3） ソフトウェア信頼度成長モデルの評価

ソフトウェア信頼度成長モデルの適用性を示すために，推定値に対する実測値の相対誤差を用いて，ソフトウェア信頼度成長モデルの推定精度がいかに現実に即しているか検証を行う.

検証を行うにあたって，以下のように記号を定義する.

T ：今回のテスト期間

A_t ：テスト進捗率 t で推定したソフトウェア内に潜在する推定総フォールト数

N_T：時間区間(0,T]の間に報告されたフォールト数

G_t ：相対誤差(%) $=(|N_T - A_t| / A_t)\times 100$

すべての機能が実装された状態でテストが行われたプロジェクト1と，機能を随時追加していく形でテストが行われたプロジェクト2を用いたフォールト数の推定誤差を図 2.27 に示す．ここで，プロジェクト1は，図 2.24～図 2.26 に示した事例プロジェクトである.

図 2.27 より，両プロジェクトともテスト進捗率 60%から推定精度は安定しているのが分かる．プロジェクト1の場合の誤差変動をみると，テスト進捗率が 30%以降で相対誤差が小さい値で推移し，推定精度は常に安定している．プロジェクト2の場合はテスト進捗率 40%の時点で大きく上向いている．これは，機能を随時追加していく形でテストが行われたというテスト方法が影響していると思われる.さらに，プロジェクト1は，テスト項目の漏れを無くするために，品質指標を用いてユーザ要求を品質特性で分類し，テストケースの洗い出しを行うといった方法を施行した上で，成功したプロジェクトである．この点も，モデルの推定値に大きく影響していると思われる.

このように，テスト進捗の早い段階でフォールト総数が予測できるならば，その予測値を基にしてテスト工程での製品品質予測も可能となる．その結果，出荷時期を予測するマネジメントが可能となる.つまり,ソフトウェア信頼度成長モデルは，定量的な品質管理に有効であるばかりではなく，テスト工程のマネジメントにも有効であると言える.

（4） ソフトウェア信頼性評価技術導入による効果

次に，実際にソフトウェアの信頼性評価を行った効果を述べる.

まず第1に，それまでは経験的推測の下でしかソフトウェア製品品質の把握がで

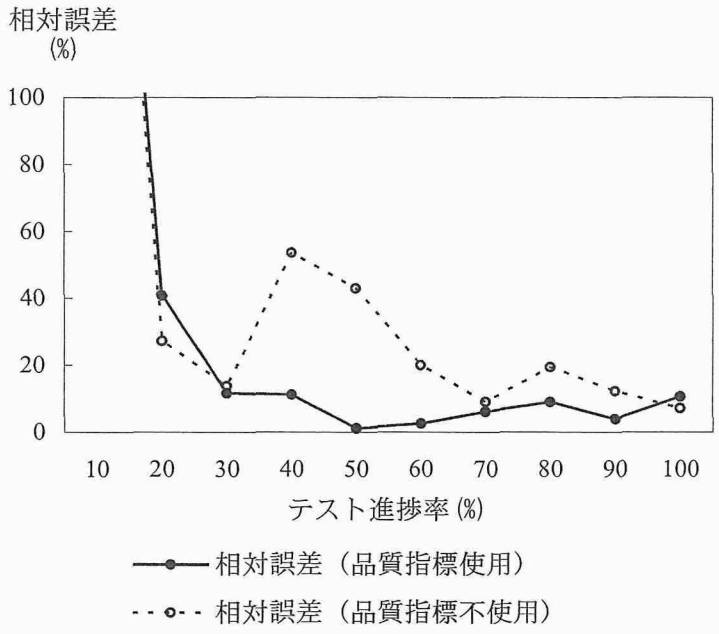


図 2.27　実測値と推定値の相対誤差

きなかったのが，定量的に表されることにより可視化できるようになり，科学的裏付けが可能となった．

第２に，出荷判定指標となるものができた．また，データ収集作業の標準化の定着とソフトウェア信頼度などの信頼性評価尺度により，ソフトウェア信頼性の達成状況や出荷品質の把握などを行うことが可能となった．

2.5.3 まとめ

人的要因が多分に影響するソフトウェア生産活動において，ソフトウェア工学に基づく品質基盤技術を導入することは，製品品質保証プロセスに次のような改善効果をもたらす．

（１） 品質保証における最後の砦として重要である製品テストにおいて，ソフトウェア品質特性を用いたテスト項目の作成により，ユーザの要求品質に着目したテストを行い，フォールトを確実に除去することができる．
（２） ソフトウェア信頼性評価技術を用いて，残存フォールトがほとんどないという製品品質レベルであることを確認することができる．

図 2.24 の事例は，品質指標を用いてテストケースの洗い出しを実施し，信頼性を定量的に評価し，その結果，顧客納品後クレームゼロの成果を得た成功プロジェクトに関する信頼性評価結果である．十分なテスト時間を費やしたことも１つの要因であろうが，ユーザ要求から外れたテストをしていては，どんなに時間をかけても顧客満足は達成できない．客先の受け入れテスト段階や運用段階においてフォールトが発生するものである．この事例は，テスト項目が十分に設計され，信頼性が確実に評価されれば，フォールト見逃しが無くなることを証明している．

2.6 おわりに

本章では，ソフトウェア開発におけるマネジメント技術として，プロジェクトのリスクを識別しリスクが発生しないかどうかを追跡するリスク管理，プロジェクト計画や進捗管理といったマネジメント技術を向上させる EVM（アーンド・バリュー・マネジメント），顧客にソフトウェア製品の品質を保証するためのソフトウェア品質保証，およびソフトウェア品質到達レベルを定量的に評価するための信頼性評価技術について議論した．

高品質なソフトウェアを開発しプロジェクトを成功に導くためには，リスク管理によりプロジェクト開始時に入り込むリスクを，プロジェクトの早い時期に軽減することが有効であることを示した．開発中は， EVM 分析回数の粒度を密に行うマネジメントが QCD 目標の達成に有効であることを示した．ソフトウェア製品品質を保証するための品質保証の仕組みが重要である．さらにテスト時は，ソフトウェア品質特性を用いたテスト項目によりユーザの要求品質に着目したテストを行い，ソフトウェア信頼性評価技術を用いて品質到達レベルを評価することにより出荷後のフォールトを無くせることを示した．

高品質なソフトウェアを，顧客と約束した期間内に，組織と約束した予算で常に開発できることは，ソフトウェア開発技術者の願いである．マネジメントを適切に実施できたときに，ソフトウェア開発を成功させることができる．そのためには，早期にプロジェクトリスクを軽減し，プロジェクトの進捗とソフトウェア品質を予測・制御し，ソフトウェア製品品質を保証するプロジェクトマネジメント技術の向上は，継続して取り組んでいく大きな目標である．今後はさらに，仕様の早期確定，顧客満足の獲得，マネジメント技術の向上などのモダン・プロジェクト・マネジメントに基礎を置いて，プロジェクトのマネジメント技術を向上する PDCA サイクルを継続していく．

第3章 プロセス品質向上のためのソフトウェアプロセス改善技術

現実のソフトウェア製品の開発では，曖昧な作業計画から作業のやり直しや追加などの無駄が発生し，顧客の受入れテスト段階でもフォールトを指摘されて作り直しをする事態に陥っている．これらの無駄の多さから，品質低下・コスト超過・納期遅延といったプロジェクト失敗の結果となる．

これらの状況から抜け出して，高品質なソフトウェア製品を開発し生産性を向上させるためには,自分たちの組織にとって効果的なソフトウェアプロセスを構築し，向上させていく必要がある．

ソフトウェアプロセスとは，高品質なソフトウェアを，開発予算で，顧客との約束の日に納入するために，プロジェクトは何をすべきか，成功する方法を考えて文書化したものである．プロジェクトマネジメント技術を開発組織に定着させるための枠組みとして必要であり，マネジメント技術の向上に合わせて改善していくものである．ソフトウェア開発が継続的に成功するようになるために，常にソフトウェアプロセスを改善し続けることが重要である．

本章では，ソフトウェアプロセスの考え方と改善方法について述べる．また，ソフトウェアプロセスモデル CMM (Capability Maturity Model，ソフトウェア能力成熟度モデル)[8],[9]の導入方法とその有効性について示す．さらに，実際に構築したソフトウェアプロセスの事例を示す．

3.1 ソフトウェアプロセス

高品質なソフトウェア製品を開発するためには，プロジェクトの最初の要求分析から，設計，作成にいたるまでのソフトウェア開発プロセスで品質を作りこみ，テストプロセスで品質を確認する仕組みが必要である．各プロセスを切り離して考えたり省略したりすれば，プロジェクトは必ず失敗する．本節では，ソフトウェア開発の入口から出口まで，どのようなソフトウェアプロセスが良いのか，どのようにすればソフトウェアプロセスを良くできるのかについて議論する．

3.1.1 ソフトウェアプロセスの定義と考え方

ソフトウェアプロセスとは，端的に言うと，ソフトウェアの作り方である．作り方が良ければこそ，作られるソフトウェア製品の品質も良いといえる．ソフトウェアプロジェクトは，高品質なソフトウェア製品を開発予算内で顧客との約束の日に

渡すことができるための手順として，具体的にソフトウェアプロセスを決めることが重要である．プロジェクトマネージャは，このソフトウェアプロセスに基づいてプロジェクト計画を立案し，その進捗状況を管理していくことにより，プロジェクトを成功に導くことができる．

なお，CMM では，ソフトウェアプロセスをソフトウェアとこれに関連する成果物（計画書，設計書，コード，テスト書，マニュアルなど）を開発し保守するための活動および方法論と定義している．

以下に，CMM におけるソフトウェアプロセスの基本的な考え方を列挙する．

- よい品質はよいプロセスから．
- プロセスの省略が失敗への道．
- マネジメントのプロセスがなければ，ソフトウェア開発のプロセスは活きない．
- 個人のプロセス，プロジェクトのプロセス，組織のプロセスの階層で考える．
- プロセスをよくするには，プロセスを文書で定義し，発生した問題は2度と発生しないよう，PDCA の管理サイクルを回していく．

3.1.2 ソフトウェア開発プロセスの改善事例

ソフトウェア開発プロセスとは，顧客との契約に始まり，要求分析，設計，コーディング，テスト，出荷までの一連の開発プロセスのことである．ソフトウェア開発プロセスを改善する事例として，要求分析・定義プロセスについて示す．

（1） 要求分析・定義プロセス

顧客要求を分析し定義するために必要な技術を下記に示す．

- 顧客要求を要求仕様書に文書化し，完成イメージを見えるようにする技術
- ソフトウェアを使用する現場を調査し把握する技術
- 完成イメージを設計工程に伝える技術

最近のソフトウェア開発では，その要求分析・定義技術の低さにより，その後の開発で品質と生産性を低下させてしまうプロジェクトの失敗事例が多い．経験した失敗の原因としては，プロジェクトの契約範囲が不明確なまま作業を進めたり，仕様の定義を見落したり，顧客の運用業務を理解できていなかったり，またそのような状態を放置しておくことなどが原因であった．

この改善策として，プロセスの最初の成果物である要求仕様書の作成時には，顧客要求に曖昧性がなく，すべて網羅されていることをレビューで確認するよう改善した．その後，設計からテストまで顧客要求が首尾一貫して伝えられていることを

レビューで追跡するよう改善した．テストの終了時には，プロセスの最後の成果物であるソフトウェアが顧客要求どおりのソフトウェアであることを評価するよう改善した．このような仕組みをもつことにより，失敗を大きく減少させるプロセスに改善することができた．

実際に要求分析・定義プロセスを改善するには，プロセスの最初の成果物である要求仕様書の利用目的を理解して取り組んだ．要求仕様書は，顧客との合意文書であり，設計プロセスおよびテストプロセスと保守プロセスへの入力文書である．したがって，要求仕様書は，顧客の視点で記述されており，さらに漏れなく機能仕様と品質仕様について定義され，テストが可能な文書でなければならない．

この利用目的から，要求分析・定義プロセスを次のように製品品質保証プロセスでレビューすることにより，顧客要求の曖昧性を排除し，ソフトウェア開発の入口から出口まで顧客要求を伝達する仕組みをもつことができた（図 3.1 参照）．

① 要求分析プロセスは，要求仕様書が顧客と合意されテストが可能であることをレビューによって確認する．要求仕様書レビューで使用するチェックシートの実例を表 3.1 に示す．

② テストプロセスは，開発の後工程として扱うのではなく，要求分析プロセスの後工程として扱う．つまり，テスト項目は，要求仕様を全てテストできることをレビューによって確認する．

③ テスト結果から，製品品質を評価し出荷品質として十分であることを判定する．

（2） 製品品質保証プロセス

ソフトウェア開発プロセスの改善と同期して，製品品質保証プロセスも改善した．製品品質保証とは，顧客にソフトウェア製品の品質を保証しますと言えることが目的である．下記に示すような仕組みが必要である．

- 要求品質を漏らさない仕組み
- 成果物が常にレビュー可能であるために必要な成果物定義
- 要求仕様を実現するための設計であることを確認するプロセス
- テスト項目に要求品質を反映できる仕組みとその確認プロセス
- ソフトウェアが完成したことを判定するプロセス

3.1.3 マネジメントプロセスの改善事例

マネジメントプロセスとは，プロジェクトマネージャが実施するプロセスを言う．プロジェクトが QCD 目標を達成するために，顧客要求を管理し，ソフトウェア開発プロセスとスケジュールを計画し，その進捗を管理するとともに，ソフトウェア

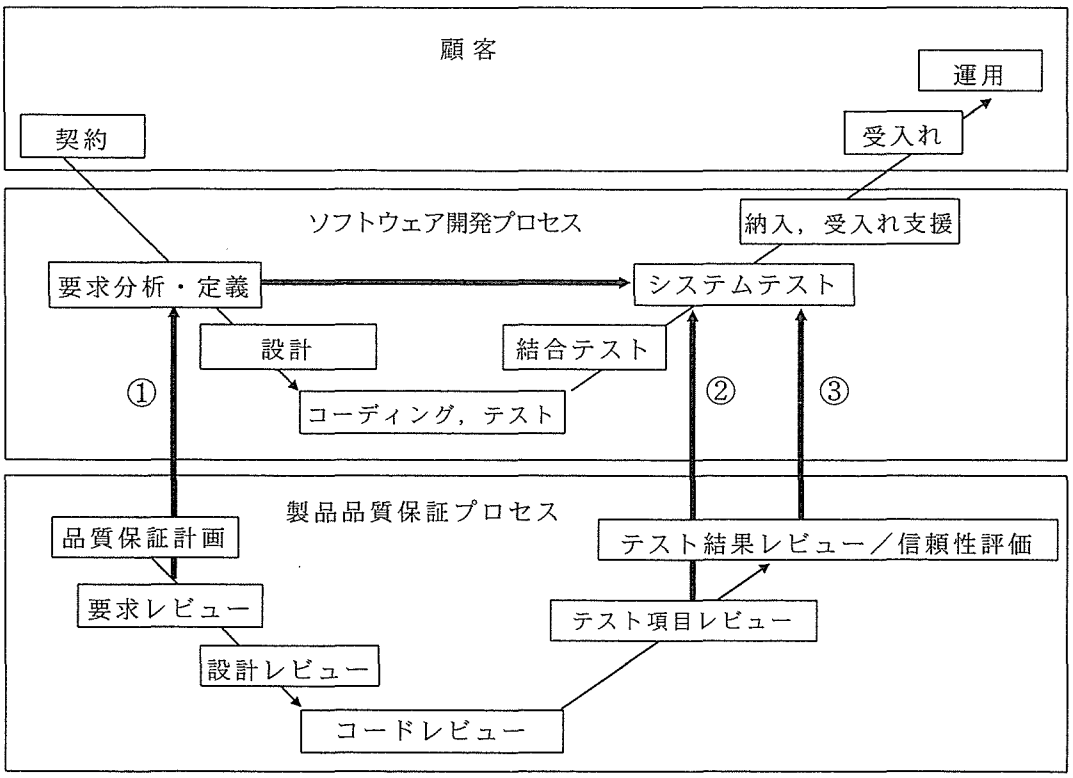


図 3.1 ソフトウェア開発プロセスの改善

表 3.1 要求仕様書チェックシートの実例

1．機能仕様		
（1）	目次	目次で全体が見える
（2）	概要	要求の要旨を抽象化，モデル化している
	ユーザ購入動機	要求のわけを記述している
		現状業務の問題点を記述している
（3）	ユーザ業務	顧客の業務を理解している
		顧客の要求を引き出している
	業務フロー	業務で扱う物を定義している
		業務で流れるデータを定義している
（4）	システム構成と受注範囲	業務フローではシステム化範囲が明確である
		ソフトの対象範囲が明確である
		ハードの対象範囲が明確である
	システム制限事項	システムの制約条件が明確である
	装置	装置仕様を記述している
		装置の流れと境界定義をしている
（5）	機能　　概要	機能一覧と機能概要を記述している
	機能毎	機能毎に詳細化している
（6）	ユーザインターフェイス	運用がイメージできる
2．品質仕様		
（1）	性能（MAX量）	最大値を定義している
		異常ケースの対応を決めている
		要求は検証できる
（2）	性能（処理時間）	応答時間，処理時間の要求性能を記録している
		要求性能は検証できる
（3）	信頼性	ハード障害時のシステム保証の範囲を記述している
		ソフト障害時のシステム保証の範囲を記述している
		（検出，報告，再試行，縮退）
		実運用での安定動作を証明できる
（4）	操作性	機能の重要度，使用度，使い勝手の要求を記述している
（5）	保守性	故障発生時の一次切り分け実施方法を記述している
（6）	移植性	再利用範囲の定義をしている
（7）	安全性	PL問題事項とみなされる項目を識別している
		セキュリティレベルを定義している
3．作業仕様		
（1）	成果物	仕様書のマイルストーンと納期を確認している
		設計書のマイルストーンと納期を確認している
		試験成績書のマイルストーンと納期を確認している
		コードのマイルストーンと納期を確認している
（2）	開発環境	開発ハードの種類と必要期間を記述している
		OSとツールの種類を記述している
（3）	テスト環境	検証ハードの種類と必要期間を記述している
		検証ツールの種類と必要期間を記述している
（4）	受入検証	顧客の受入検証期間と基準を確認している
（5）	現地調整	期間を記述している
（6）	品質保証（瑕疵）	期間と範囲を記述している
4．文書確認		
（1）	仕様未決定事項	暫定案を作っている．暫定であることを区別している
		（後日決定は不可）
（2）	ドキュメンテーション	箇条書きにして番号を付けている．シンプルな文である
（3）	完成イメージ	完成イメージを顧客に見せれる
		完成イメージを次工程につたえられる

開発プロセスの品質を保証するプロセスのことである．マネジメントプロセスを改善する事例として，プロジェクト計画プロセスと進捗管理プロセスについて示す．

（1） プロジェクト計画プロセス

プロジェクトマネジメント技術の低さにより，スケジュール超過や品質に影響を及ぼしてしまう失敗事例が後を絶たない．経験した失敗の原因としては，ソフトウェアの開発規模を推定できていないにもかかわらず，顧客の要求納期から逆算してスケジュールを作ったり，実現見込みのない要求納期を受け入れたりなどがあった．さらに，マネジメントをしないで開発作業を担当しているプロジェクトマネージャや，開発者のスキルを把握していないプロジェクトマネージャも実在した．プロジェクトを失敗させないためには，このような多くのマネジメントリスクに対応していかなければならなかった．

この改善策としては，プロジェクト計画をレビューし，さらに開発中は，スケジュールの進捗を管理し計画精度を向上させていく仕組みをもつよう改善した．この仕組みにより，ソフトウェア開発における失敗原因の過半数を占めるといわれるマネジメントの失敗を大きく減少させるプロセスに改善することができた．

実際にマネジメントプロセスを改善するには，プロジェクト計画と進捗管理の内容を理解して取り組んだ．PSP(Personal Software Process) [13]によると，「プロジェクト計画は，作業とその実施方法を定義したものである．主要タスク毎の定義，それに要する時間とリソースの見積り，管理者によるレビューと制御の枠組みを与える．適切に文書化すると，実際の遂行能力と比較するためのベンチマークとなる．この比較によって，計画者は見積りの誤りを理解し，見積りの正確さを改善できる．」と定義されている．よって，プロジェクト計画の内容には，開発規模，開発の進め方，進捗を管理する仕組みを定義しなければならない．また，開発が終了したときに評価のできる計画を作ることが重要である．そのためにプロジェクト計画の立案では，以下のことを行う．

- 開発規模
  ソフトウェアの開発規模や期間を見積る．
- 開発の進め方
  開発作業を進める段取りとして，プロジェクトのプロセスを定義する．
- 進捗を管理する仕組み
  どこまでプロジェクトが進んだかを知り，期限までに終了できるかどうかを管理するための詳細スケジュールを作成する．
- 事後評価できるための根拠
  計画精度はどうだったか，今回の課題は何か，次回の改善策は何かを評価するための仕掛けを作成する．

図 3.2 に，プロジェクト計画立案の枠組みを示す．プロジェクト計画を立てる順序は，まず顧客要求を定義する．次にソフトウェア開発プロセスを定義することにより開発の進め方を決定する．その上で開発規模を算出し，リソースを見積る．リソース見積りでは開発者のスキルの見積りが重要である．スケジュールは，これらのすべての情報を入力として組み立てる．

（2） 進捗管理プロセス

進捗管理の目的は，プロジェクト計画と照合することにより，計画と現在の進捗状況との差異原因を分析し，プロジェクトが今後どのように推移していくのか，QCD の目標値を予測することである．そのために進捗管理では，以下のことをおこなう．

- 計画した開発規模の実績値を測定する．
- スケジュールの実績値を測定する．
- 計画値と実績値の差異を監視する．差異が管理限界を超過すればマネジメントリスクの発生と認識し，原因を分析する．
- 問題があれば，対策を考えて再計画する．

（3） 品質保証プロセス

以上の観点のマネジメントプロセスに対して，品質保証プロセスにて監視を行うことにより，マネジメントスキル不足による失敗を防止し，常に目標値を予測し計画精度が高められるようにリスクに強いマネジメントプロセスとすることができた．品質保証プロセスでは，図 3.3 に示すように，マネジメントプロセスに関して以下の確認をしている．

① プロジェクト計画では，ソフトウェアの開発規模が見積もられ，作業の進め方と成果物が定義され，スケジュールの進捗管理ができるように詳細に分解されていて，成果物も全て決められていることなどをレビューで確認する．プロジェクト計画レビューで使用するチェックシートの実例を表 3.2 に示す．

② 進捗管理では，定期的に実績値を計測し，計画値との差異を分析する．管理限界を超える場合は，その原因を分析し再計画する．再計画後は遅れが拡大しないように追跡し，再計画の有効性を評価する．

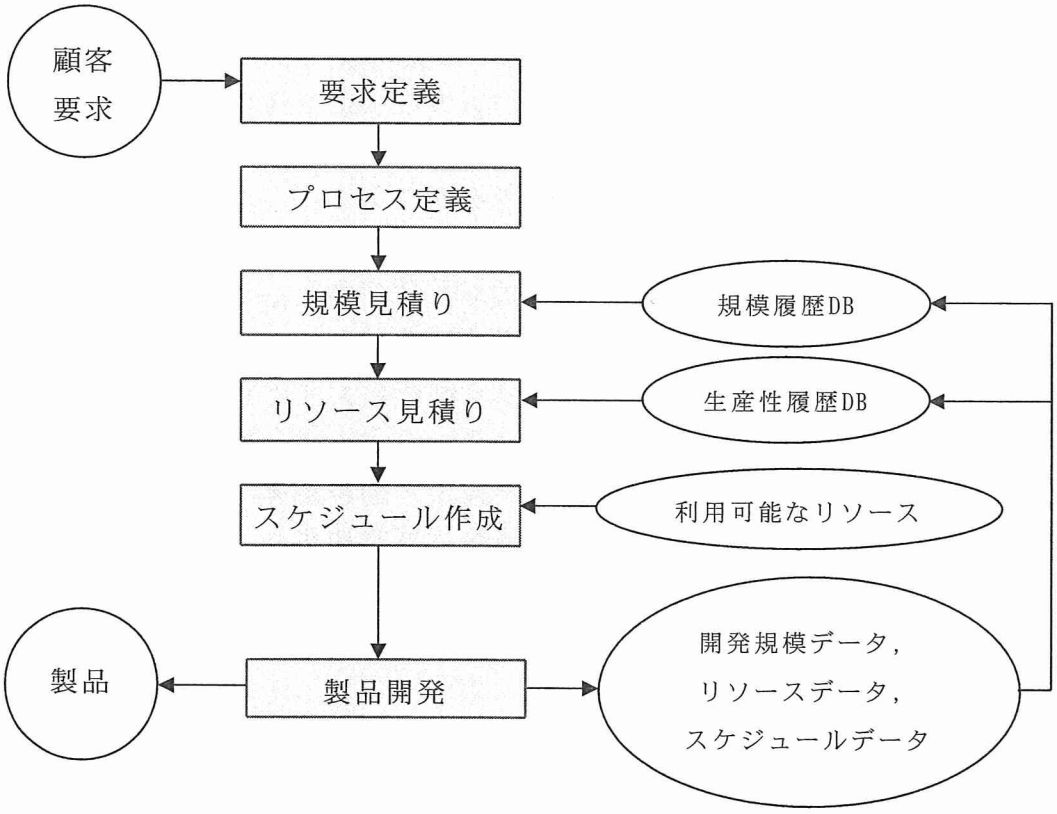


図 3.2 プロジェクト計画立案の枠組み

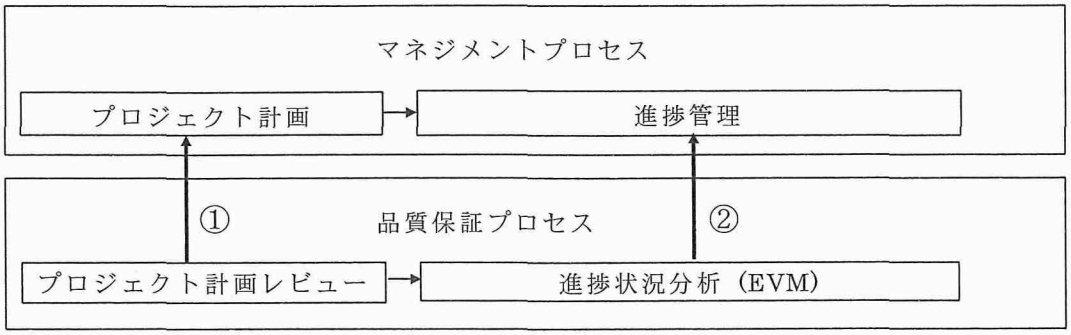


図 3.3 マネジメントプロセスの改善

表 3.2 プロジェクト計画チェックシートの実例

	質問事項
規模見積り	①作業成果物を明確にしているか ②見積りが行えるレベルに作業成果物を細分化しているか ③過去のプロジェクトを参考に活用しているか ④見積りに対する制約・前提条件を明確にして文書化しているか（例：ハードの提供が条件） ⑤見積りを文書化し，レビューを実施し，プロジェクトに関するメンバーの同意を得たか
工数・費用	①規模見積りに基づき，工数・費用を見積もっているか ②過去のプロジェクトを活用しているか ③見積りを文書化し，レビューし，関連するメンバー（社内及び顧客）の同意をえているか
リスク	①プロジェクトのリスクが分析されており，その対応策が文書化されている ②過去の失敗原因に対した改善策がプロジェクトに盛込まれている ③メンバーのスキルをPMが把握しているか
プロジェクト	①プロジェクトに関わるメンバーの役割と責任（PM，リーダー，メンバー，営業，品質保証など）を明確に文書化し，顧客の承認を取ったか ②メンバーの変更管理が行われているか
開発環境	①開発環境・テスト環境といった必要なハード・ソフトを考慮しているか ②調達・管理する責任者を決めているか ③開発環境を必要とするメンバー（開発メンバーや，品質保証メンバー）で，開発環境の計画をレビューしているか
スケジュール	①見積もりから展開されたスケジュールとなっているか ②顧客とのレビュー，社内レビューなどのマイルストーンや，作業間の依存関係（ハードがないとテストができない，基本設計後でないと詳細設計できないなど）をスケジュールに反映させているか ③進捗を正確に測定できるように，各作業は10人日以下のボリュームとなっているか ④作業に対する成果物が明確で，100%達成を確認できるか ⑤スケジュールの前提条件（ハードが無いとテストできないなど）を明確に記述しているか ⑥スケジュールを文書化して，レビューし，関連するメンバー（社内，顧客）の同意を得ているか

3.2 ソフトウェアプロセスモデル CMM

3.2.1 CMM

米カーネギー・メロン大学ソフトウェア工学研究所(CMU/SEI)が作成したCMM(Capability Maturity Model)[8]は，ソフトウェア開発プロセスの管理能力の成熟度を測る“ものさし”であり，同時にソフトウェアの開発組織における，プロセス改善作業の優先順位の指針となる．CMM を使えば，ソフトウェア開発組織の状況を評価し，その組織に合わせたプロセス改善の目標を設定することができる．

ここで，ソフトウェア開発プロセスの管理能力とは，ソフトウェア開発プロジェクトがどのように進んでいくかを予測する能力のことである．CMM は，ソフトウェア開発プロセスがどれだけ定義され，管理され，測定され，制御され，効果的であるかを測る尺度である．

CMM の考え方は，「ソフトウェア開発組織は一定の段階を経て発展していくものであり，その発展段階に応じた改善活動を行うことが，プログラムの品質や生産性の向上に有効である」[9]というものである．個々のソフトウェア開発組織が抱える問題は同じではない．それぞれの組織に合わせて，プロセスの改善を実施することが重要である[7]．我々の経験プロジェクトにおけるプロセス改善も，期待する改善効果と因果関係の強いプロセスに焦点を当てて取り組んだ．

（1） プロセス成熟度とは

プロセス改善に取り組む前のソフトウェア開発組織には，以下のような事例が多く見られる．

- プロセスはその場しのぎで定義され，ソフトウェア製品の品質を判断する客観的な基準が存在しない．
- ソフトウェアが顧客に納入されるまで，どのようなものが作られるかわからない．

それに対して，CMM では，開発組織の状況にふさわしい取り組みを提案している．例として，“設計内容のレビューが行われなかった”という問題に対し，組織は成熟度によって次のように対応が異なる．

- レベル１の組織（プロセス改善に取り組む前の組織）の改善策
  設計書や要求仕様書は書かれたらレビューすることを約束する．
- レベル２の組織（プロセス改善に取り組んだ組織）の改善策
  設計書の標準的な目次やテンプレートを確認する．レビューが計画され進捗管理が実施されているかどうかを確認する．

上記の例は，レベル２の成熟度の組織とは，プロジェクトマネジメントができる組

織であるということを示している．

（2） ソフトウェア開発プロセス成熟度の5段階

成熟度段階とは，より成熟したソフトウェア開発のプロセスに到達するように定義された段階的なステップのことである．以下の5段階がある（図3.4参照）．

① レベル1

初期プロセス(initial process)：プロジェクトの予測がつかず，失敗が多い

② レベル2

反復できるプロセス(repeatable process)：プロジェクトマネジメントができるレベル

③ レベル3

定義されたプロセス(defined process)：レベル2のプロジェクトマネジメントを組織運営できるレベル

④ レベル4

管理されたプロセス(managed process)：組織的なプロセスを定量的に管理できるレベル

⑤ レベル5

最適化しているプロセス(optimizing process)：定量的な計測を用いて，プロセス改善できるレベル

レベル1を除き，それぞれの成熟度段階は，いくつかのKPA（キー・プロセス・エリア）に沿って判断する．KPAとは，組織がソフトウェア開発プロセスを改善する際に焦点を当てるべきもので，ある成熟度段階に到達するために満足すべき項目である．

プロセスは段階的に改善すべきものであり，組織はプロセスを安定して反復できるようになって始めて，プロセスの改善に本格的に取り組むことができる．個々の

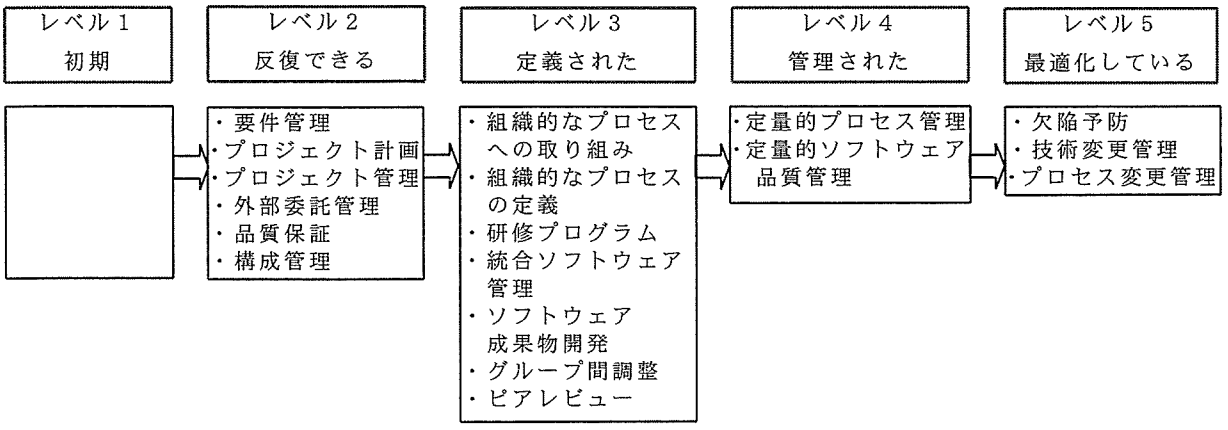


図3.4 プロセス成熟度の5段階と成熟度段階ごとのKPA

ソフトウェアの開発組織が抱えている問題は同じではない．それぞれの組織に合わせてプロセスの改善を実施することが重要である．

図 3.5 は，レベル１からレベル３までの組織におけるソフトウェアプロセスの可視性の違いを表している．レベル１では，プロセスがないため終了まで結果を見ることはできない．レベル２では，開発フェーズ単位にプロセスが定義され計測されるので，開発フェーズ単位で結果を見ることができるようになる．さらにレベル３では，開発フェーズの内容が定義される．この結果，次のレベル４の定量的な品質管理が可能となる．

図 3.6 は，レベル１からレベル３までの組織における QCD 目標の達成確率を表している．レベル１では，スケジュールやコストの実績値が目標を大きく超過する．レベル２では，目標が現実的となり，スケジュールやコストに関する目標の達成確率は高くなる．さらに，レベル３では，後戻り作業などのリワークが減り，プロジェクトは目標内に完了できるようになる．

（３） IDEAL アプローチ

CMU/SEI が開発したプロセス改善への取り組み方法は，以下に記す５つのフェーズの頭文字をとって，IDEAL アプローチと呼ばれている．各フェーズは以下のとおりである．

① 開始フェーズ(Initiating)
改善活動体制の確立のため，SEPG(ソフトウェア・エンジニアリング・プロセス・グループ)を組織し，ソフトウェア開発プロセスに関する知識を身に付ける．

② 診断フェーズ(Diagnosing)
現在のソフトウェア開発プロセスの状況を認識し，文書化する．現時点の組織のベストプラクティスと，CMM に定義されているキープラクティスを比較して，目標となるプロセスを設定し，文書化する．

③ 構築フェーズ(Establishing)
目標プロセスの取り組み優先順位を決定する．ワーキンググループを作って，活動を計画する．

④ 実施フェーズ(Acting)
ワーキンググループは，取り組む目標プロセスをパイロットプロジェクトに適用する．

⑤ 発展フェーズ(Leveraging)
パイロットプロジェクトの評価を文書化し，組織的なアプローチを変更し，本運用する．

各フェーズは成熟度レベルに基づいて評価され，その手段としては，SPA（ソフ

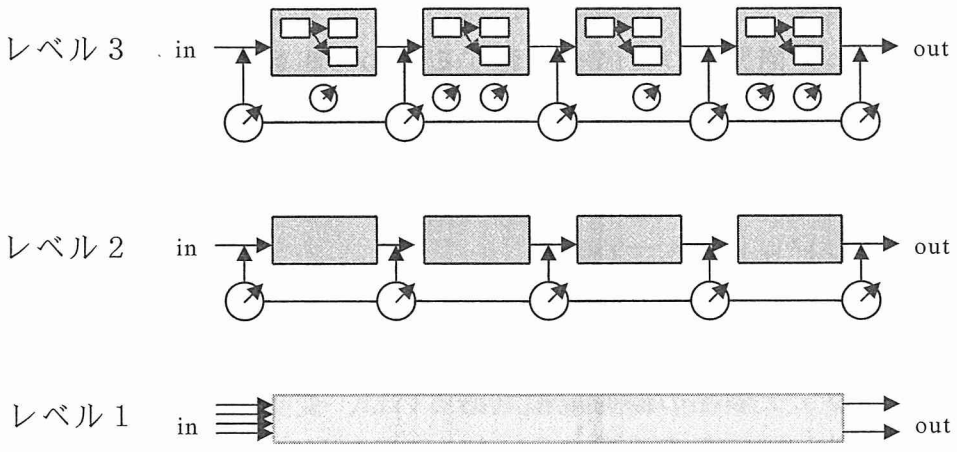


図 3.5　ソフトウェアプロセスの可視性

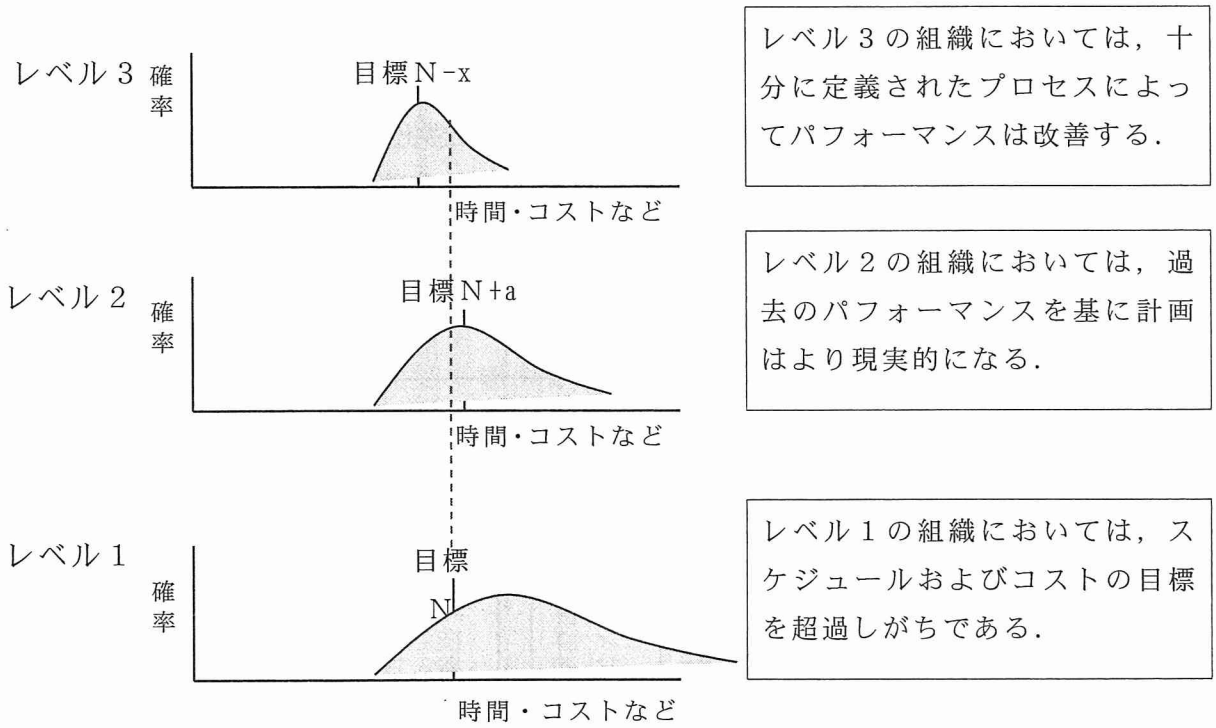


図 3.6　成熟度段階ごとのプロセス能力

トウェア・プロセス・アセスメント）が推奨されている．SPA は，開発組織におけるソフトウェア開発プロセスの現状を把握し，直面する課題を分析して優先順位を付け，ソフトウェア開発プロセスを改善するための組織からの支援を得るために実施する（図 3.7 参照）．例えば，組織がレベル２以上であるかどうかを判断する SPA の質問としては，以下のようなものを用いる．

- 要求事項は実装することが可能で適切であるか，検査可能であるかなどの確認のレビューをしているか．
- プロジェクト計画のための文書化された組織の方針（ガイドライン，手順など）があるか．
- ソフトウェア製品の開発規模（あるいは，変更の規模）を把握し，見積り値と比較して，是正処置をとっているか．
- テストの検出フォールトに関する統計データを収集しているか．
- ソフトウェア品質保証の担当者は，ソフトウェア開発プロジェクトの管理者とは独立した上級管理者への連絡手段をもっているか．

これらの質問に対する回答から，満足されていない項目を明らかにして，必要な遂行計画を作る．

CMM の策定にかかわった CMU/SEI のハンフリー(W.S.Humphrey)教授によると，ソフトウェアの開発組織がプロセス成熟度レベルを上げていくためには，レベル１からレベル２への移行や，レベル２からレベル３への移行で１～３年の期間が必要だという[7].

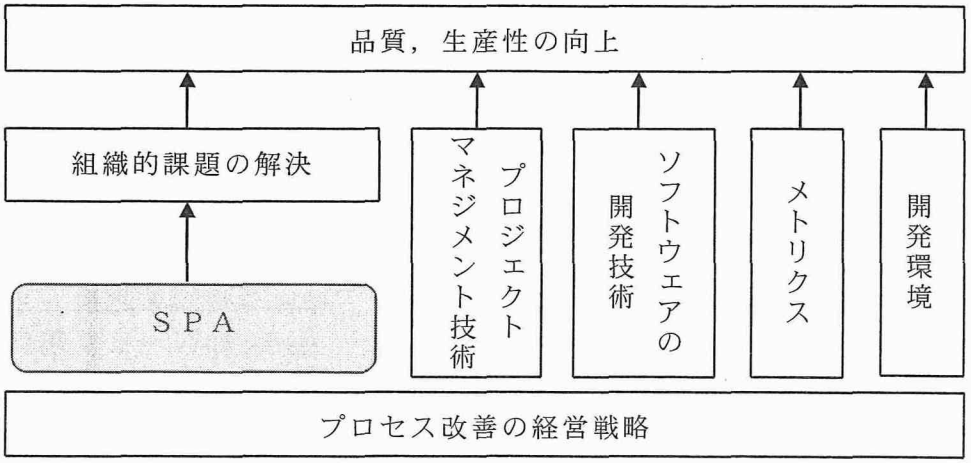


図 3.7　SPA によるプロセス成熟度の改善

（4） 品質保証と品質管理

CMM の KPA は，「コモンフィーチャ」と呼ばれる以下の５つで構成する．

- 実施コミットメント
- 実施能力
- 実施活動
- 計測と分析
- 実施の検証

特に，CMM に取り組むには，上記の「計測と分析」と「実施の検証」は不可欠なものである．

「実施の検証」とは，ソフトウェアの開発組織が確立された一定のプロセスどおりに作業を行っていることを検証するものであり，上級管理職，プロジェクト管理者，品質保証担当者による品質保証である．品質保証は，レベル２で品質保証 KPA として仕組みを構築し，レベル３の組織的なプロセス定義 KPA で完成する．

「計測と分析」とは，ソフトウェア開発プロセスを，主に開発規模，スケジュール，品質について計測し，その測定値を分析するものである[9]．しかし，計測を前提とする品質管理はレベル４にあり，定量的プロセス管理 KPA で完成する．これは，正確な計測の実施には，上記の品質保証を完成させていなければならないからである．品質管理は，品質上の目標を定めて，開発作業の結果を計測し，目標に近づけるための管理であるからである．

現実の問題として，レベル１の組織が要求仕様を漏らすなどの理由で，手戻りが多発して業績が悪化しているような場合，計測を行うことが必要である．この事例の場合，要件管理に関して計測を実施して計測結果を組織に示すことにより，組織の支援を得た上で，目標を立てて要件管理 KPA の改善に取り組むことができる．

品質管理活動は，高品質のソフトウェア製品の開発とプロセスの改善のために，品質保証への取り組み状況に合わせて最初から取り組んでいくべき活動である．

3.2.2 CMM の導入事例

ソフトウェア開発における QCD に関するプロジェクトの成功と高い顧客満足度の実現は，多くのソフトウェア開発組織の目標である．しかし，ソフトウェアプロセスの改善には相当のコストと時間が必要であり，プロジェクトに対する理解の習得などを含めて立ち上げが難しい．

改善活動は，失敗経験による再発防止としてのリスク管理活動の中で取り組んだ．リスク管理活動で得た情報を基にして，CMM を取り上げ，プロセス成熟度レベル 2 を目標にプロセス改善指針を作成し，本格的なプロセス改善活動に発展させたのである．CMM の導入理由は，組織としてのソフトウェア開発能力を順次向上できる

柔軟な方法であり，目標の達成度や達成のための手順や方針を都度見極めることができる点を考慮した．

（１） CMM の導入方法

ただ漠然と CMM を当てはめて，改善活動を進めればよいというものではない．この場合には問題定義は容易だが，改善項目が様々なプロセスにまたがり必然的に多くなる．これらを短期間に順次対応するのは困難であり，改善にかかるコストも必要以上に多くなる．そこで，改善効果と強い因果関係にあるプロセスに焦点を当て，改善策に優先順位を設けることにより，プロセス改善の有効性と問題点を，早期に確認できるようにした．

初年度は，プロジェクトマネジメントの標準的な仕組みを構築するという目標のもとに，CMM レベル２で定義している「要件管理」，「プロジェクト計画」，「プロジェクト管理」，「品質保証」，「外部委託管理」の各 KPA を下記の方法で順次導入した（図 3.8 参照）．

① CMM をガイドラインとして目標プロセスを設定し，文書化する．
② 現状プロセスと目標プロセスを比較し，現状のプロセスに不足している部分を目標プロセスで補った新プロセスを定義する．
③ 新プロセスを導入したパイロットプロジェクトを実施する．
④ パイロットプロジェクトでの効果を確認した後で，新プロセスをリスク管理活動の全対象プロジェクトで運用する．

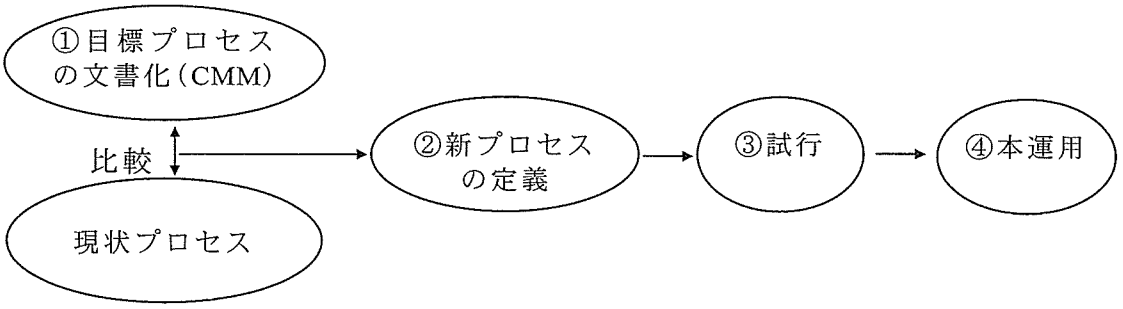


図 3.8　CMM の導入方法

（２） 目標プロセスの設定で用いたマネジメント技術

プロジェクトマネジメントの標準的な仕組みを構築するにあたり，目標プロセス設定のために関連付けたマネジメント技術を以下に示す．

・リスク管理

リスク計算書によりプロジェクトの失敗をリスク度として評価し，リスク

対策を施す管理技術.

・要件管理
顧客の要求を引き出し，顧客と顧客要件について共通の認識を得るための管理技術.

・プロジェクト計画
作業とその実施方法を，プロジェクト体制や進捗状況がわかるよう定義する技術.

・進捗管理
作業の進捗を管理し，計画を外しそうな兆候を察知し，軌道修正する技術.

・レビュー技術
設計品質計測シートを用いて，成果物の誤りや欠陥を検出・確認し，評価する技術.

・信頼性技術
ソフトウェア信頼度成長モデルを用い，ソフトウェア製品品質を定量化し信頼性を評価する技術.

（3） パイロットプロジェクトの実施

パイロットプロジェクトによる試行において，要求定義プロセスの改善は，顧客とソフトウェア開発プロジェクトとの間で，要件について共通認識を得ることを目的に取り組んだ．改善ポイントは，以下の3つである.

・CMM で定義した要件記述書事項を基にして，契約前に顧客に発行する見積仕様書の内容を改訂する.

・契約後，技術部門がリスク管理活動に発行するリスク計算書を使ってリスクを抽出する.

・要件管理チェックシートを使って，顧客要件の管理項目，方法を定義する.

以上の改善により，ソフトウェア開発における機能要件，品質要件，契約要件に関する顧客との共通認識を深める仕組みが構築できた.

また，プロジェクト管理では，開発時の進捗モニタシートと完成日を予測するスケジュール管理シートを作成した．このシートを用いてプロジェクトの計画値と実績値を照らし合わせ，プロジェクトの今後の見込みや，QCD の目標値を予測することに取り組んだ．品質保証では，開始時に品質保証計画書を作成する．レビューでは，設計品質計測シートを用いて仕様書，設計書，テスト項目などの成果物の品質を定量化する．システムテストでは，信頼性評価を行う．この仕組みで品質を保証する.さらに外部委託時には,外部委託管理シートによるチェックの実施など,CMM レベル2をガイドラインとしたフォローの仕組みを取り込んで行った.

本運用を開始するにあたっては，リスク計算書に要件管理 KPA チェック項目およ

びプロジェクト計画 KPA チェック項目を盛り込む改訂を実施した．これは，プロジェクト開始時のリスクレビューで，技術部門において作成される見積仕様書または要求仕様書や開発計画など，要件管理項目とプロジェクト計画項目に漏れが無いかを確認する仕組みである．

3.2.3 CMM 導入後のソフトウェアプロセス

CMM 本運用後は，「全社的に QCD に関する指標の予実差を 0 にする」ことを目標として，標準ソフトウェアプロセス（図 3.9 および図 3.10 参照）の定義に基づく活動を実施した．このソフトウェア開発プロセスの標準は，レベル 3 の KPA である「組織的なプロセスの定義」をガイドラインとして，ソフトウェア開発組織における人の役割とプロセスを定義している．この定義に沿って，品質管理部門の品質保証，プロセス管理，品質管理，プロセス改善の各グループが，技術部門をフォローする．

CMM においては，標準ソフトウェアプロセスの定義は，より高いレベルの成熟度を目指すためには必須である．CMM によるプロセス定義のアプローチを支える基本的な概念は，プロセスも製品の開発や保守と似たような方法で開発し保守できるというものである[9]．定義した標準ソフトウェアプロセスも，組織の共通的なソフトウェアプロセスを確立させるために，改善の優先順位に合わせて開発してきたものである．標準ソフトウェアプロセス定義に基づく品質管理部門の活動概要は，以下に述べる通りである．

品質保証グループでは，開発グループのＶ字型モデル（図 3.9 参照）に即して，次のレビュー活動を実施する．

- プロジェクト計画のレビュー後に，品質保証計画を技術部門と合意する．
- 品質保証計画に基づいた要求仕様レビュー，設計レビューを実施する．
- テストでは，ソフトウェア品質特性（ISO9126 準拠）[5]で分類したテスト項目レビューを実施する．
- 検出フォールト数を基にして，ソフトウェア信頼度成長モデル[6],[7]を用いてソフトウェア製品品質を定量化する．

プロセス管理グループでは，開発グループに合わせた CMM レベル 2 の定義に基づく次の活動を実施する．

- リスク管理定例会を開催し，計画時には見積り仕様，プロジェクト計画，リスクレビューの実施，および開発中には進捗管理，外部委託管理のフォロー，あわせて品質保証グループの活動をフォローしてリスクを除去する．
- リスク管理定例会の中では，管理職とプロジェクトマネージャのスキル向上にも配慮し，マネージャの悩みを解消していくことや，品質が向上する

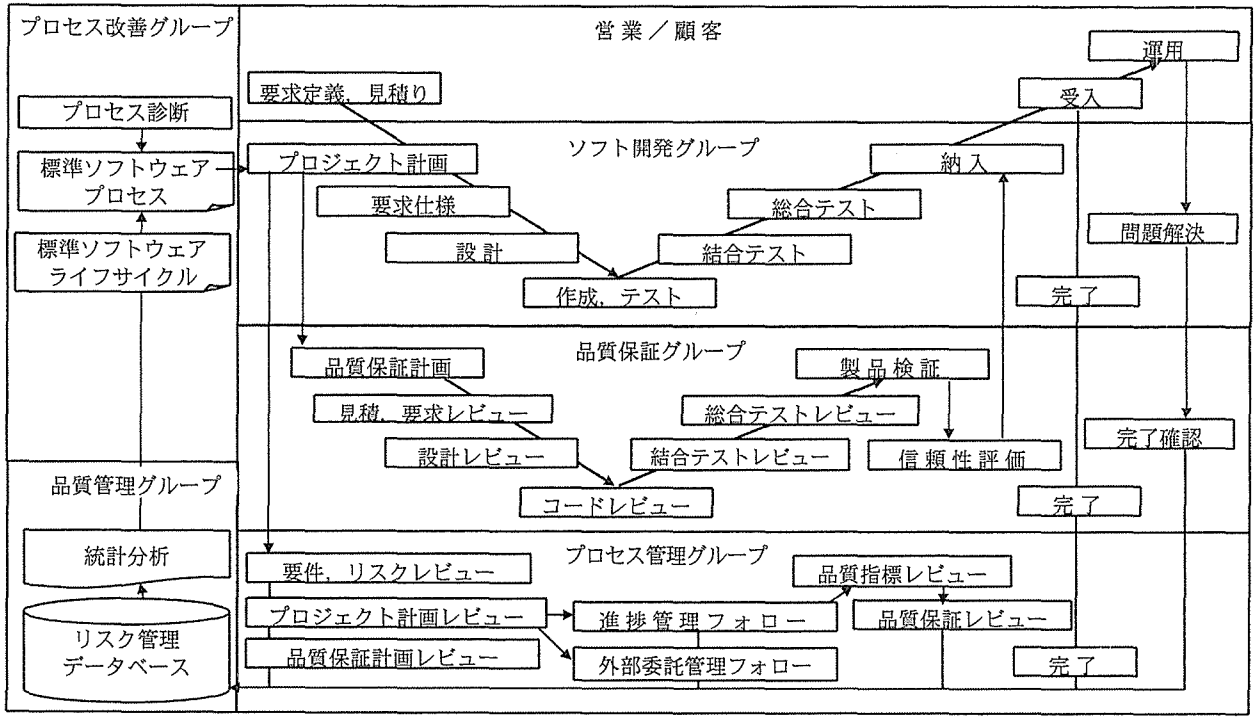


図 3.9　CMM 導入後の標準ソフトウェアプロセス

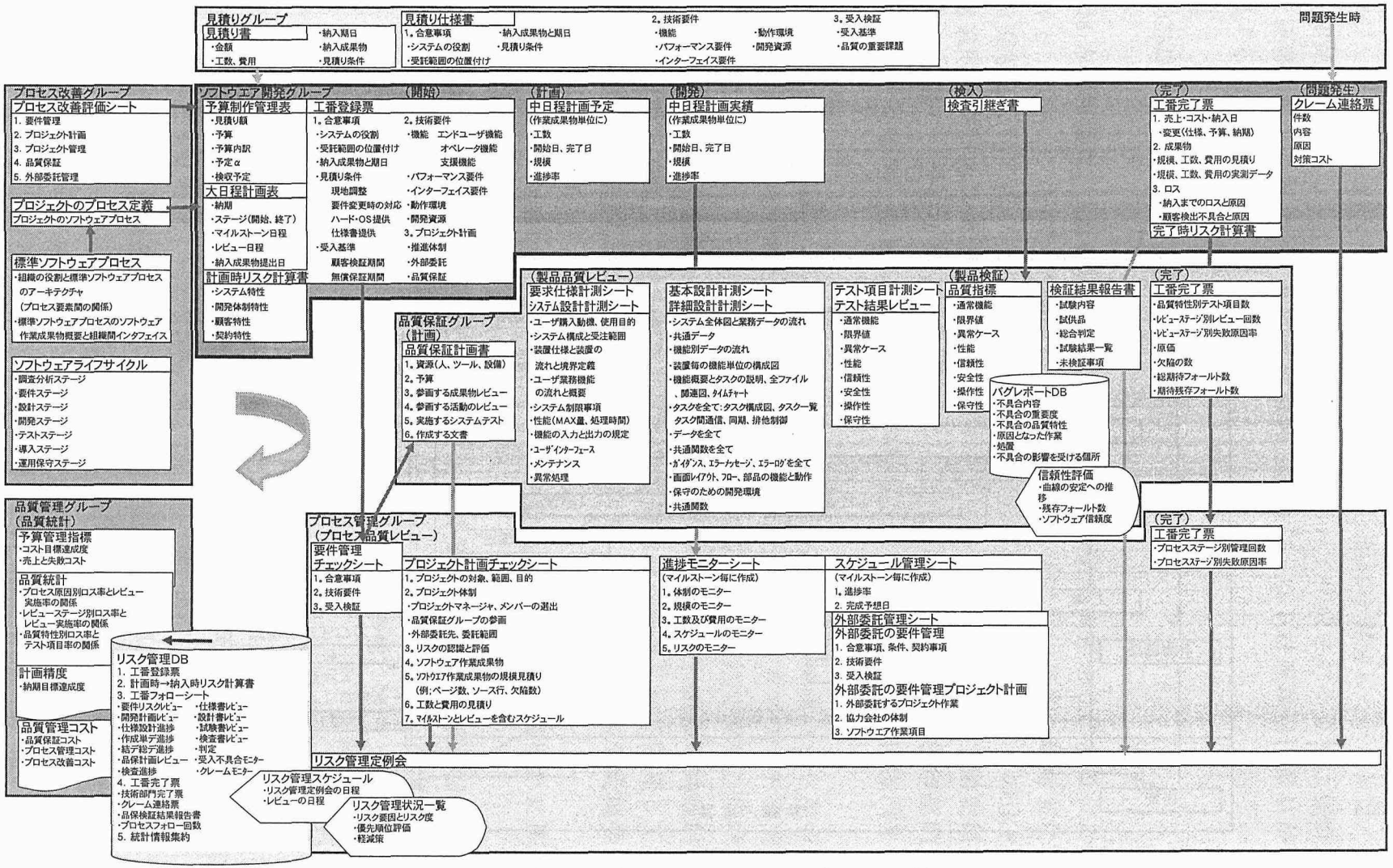


図 3.10　CMM 導入後の標準ソフトウェアプロセス詳細フロー

体験を見せていくことにより，プロセス改善を推進する．

これらの活動データを品質管理グループが受け取り，統計処理を実施して，今後の標準となるソフトウェアプロセスの改善目標に取り上げる．

3.2.4 プロセス改善の効果

プロセス改善の効果として，プロジェクトマネジメントの重要さを浸透させることができ，計画に合わせてソフトウェア作業成果物を地道に確認する習慣ができた．その結果，プロジェクトの失敗は大幅に減り，当初の大失敗を無くするという目標を達成することができた．これは，要件管理やプロジェクト計画が原因の失敗状況が，プロセス改善により短期間で改善されたことによる効果である．

図 3.11 は，プロジェクト開始時の予算に対しコスト超過したプロジェクト数の減少傾向を表している．図 3.12 は，プロジェクト毎の予算に対する超過額の減少傾向を表している．余裕・超過率とは，プロジェクト毎の予算に対する余裕額・超過額を累計した後の全売上額に対する比率である．この実績データは，過去の失敗原因と強い因果関係にあるプロセスの改善は，ソフトウェア開発組織に対して，コスト減少効果と生産性向上の即効的効果をもたらすことを示している．

3.2.5 まとめ

本研究におけるプロセス改善の特徴は，失敗から学んで再発防止を図るために，ソフトウェア開発において当り前のことをソフトウェアプロセスとして構築したことである．ツールとして使いこなせる仕組みを構築し，その仕組みに基づいてプロジェクトの成功シナリオを描き，そのシナリオに合わせてソフトウェア作業成果物を地道に丁寧に確認していくことこそが，プロジェクト成功への管理の秘訣であると考える．

CMM を利用したソフトウェア開発の改善活動において，品質管理部門が着手時に抱えている問題をどう克服していったかを以下にまとめる．

プロセス改善の提案が対象部門に受け入れられるには，品質管理部門が現場に入って技術部門の信頼を得ることが重要であった．技術者と共に問題に取り組み，プロジェクトを成功させたいという意識を共有する必要があった．立ち上げ時には，標準プロセスを押し付けることなく，対象部門のレベルに合わせてテーマを絞って取り組んだ．

さらに，技術者のマネジメント技術に対する指導や意欲への配慮をすることにより，プロジェクト管理の習慣を身に付け，ソフトウェアプロセスを浸透させることができるようになった．

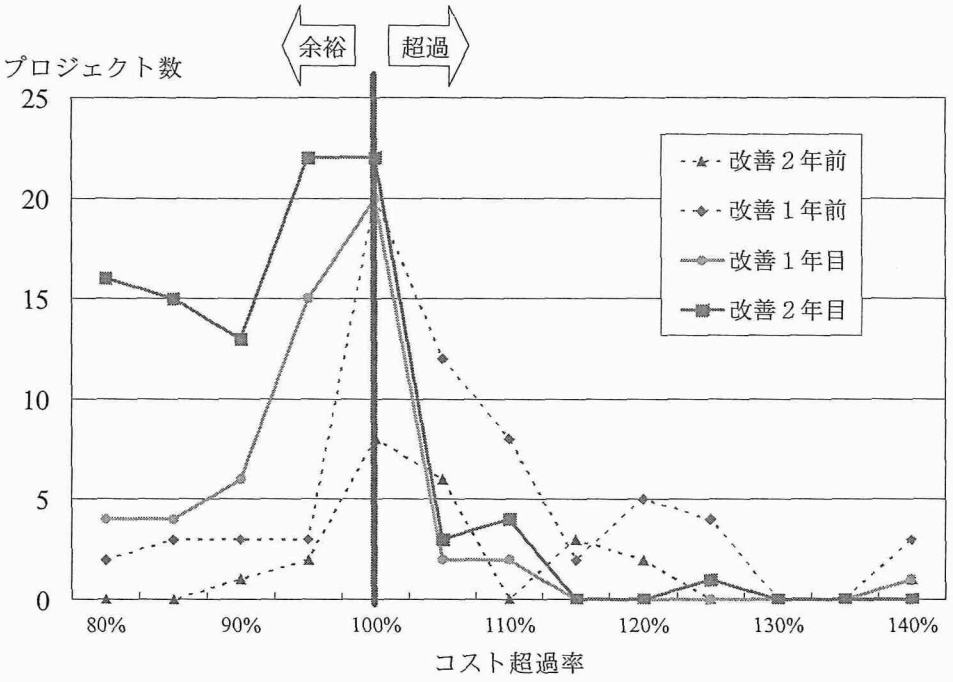


図 3.11 コスト目標超過プロジェクト数の推移

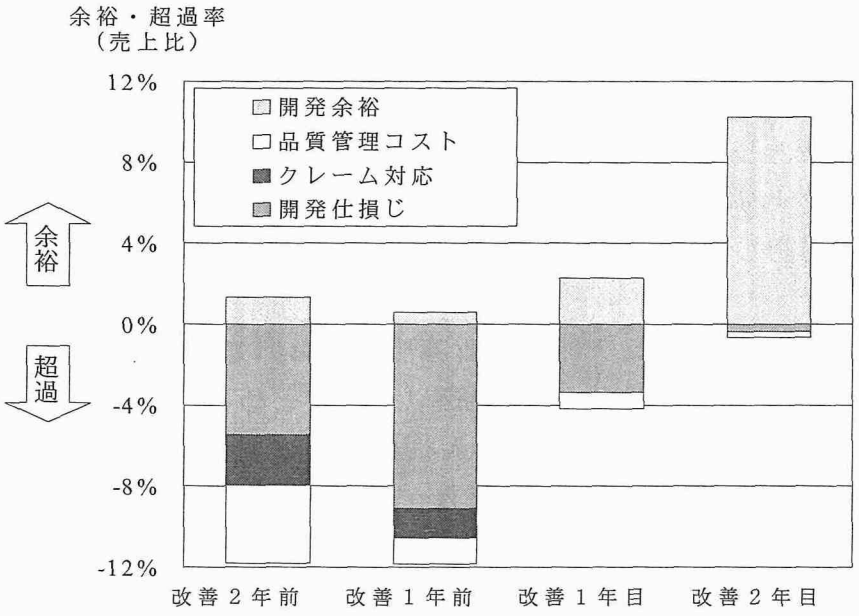


図 3.12 プロセス原因別失敗コストの推移

3.3 標準ソフトウェアプロセス

プロセス改善活動を全社的に浸透させるには，多くの困難を伴った．プロジェクトマネージャによってソフトウェア工学的な取り組みに対する理解力や実践力に個人差があったためである．そのため，全ての技術部門を同時に CMM レベル２の成熟度へ上げることはできなかった．対策として，この未熟な技術部門の課題を解決するために,標準ソフトウェアプロセスを詳細に定義し,全社的に運用を開始した．すなわち，CMM レベル２の成熟度を獲得するために，CMM レベル３の成熟度に取り組んだのである．ここで，CMM レベル３の成熟度をもつ組織とは，標準ソフトウェアプロセスを定義し運用できる組織のことである．

標準ソフトウェアプロセスとは，ソフトウェア開発が成功するためのシナリオであり，プロセスの見落しがないよう開発の仕組みを示した標準手順書である．これを基に，マネジメント技術，成果物の定義，判定基準などを向上させることにより，ソフトウェア開発プロセスの成熟度を向上させることが目的である．本節では，ソフトウェア開発が常に成功するために必要な標準ソフトウェアプロセスについて議論する．

3.3.1 標準ソフトウェアプロセスの定義

我々の経験プロジェクトにおける標準ソフトウェアプロセスは，プロジェクトの弱点として特に認識した“顧客とのインターフェイス”と“プロジェクトマネージャの役割”を，SLCP-JCF98(Software Life Cycle Process) [10]および CMM を参考にして詳細に定義している．この標準ソフトウェアプロセスの概要を，図 3.13 に示す．標準ソフトウェアプロセスは，契約プロセス，マネジメントプロセス，ソフトウェア開発プロセス，品質管理プロセスの４ブロックで構成されている．表 3.3 に，マネジメントプロセス定義の実例を示す．

この標準ソフトウェアプロセスの運用手順を下記に示す（図 3.14 参照）．

① プロジェクトの開始時にプロジェクトマネージャは，標準ソフトウェアプロセスと顧客要求からプロジェクトのプロセスを定義する．

② プロジェクトマネージャは，プロジェクトのプロセスを使用して，プロジェクト計画の作成および変更や進捗管理の一貫性を管理する．

③ 品質管理部門は，プロジェクトのプロセスが定義され，定義どおりの実践がなされているかを確認する．

④ プロジェクトの完了時にプロジェクトマネージャはプロジェクトのプロセスの実績を報告する．

⑤ 品質管理部門は，定期的に標準ソフトウェアプロセスの実績点検を行い，継続的に標準ソフトウェアプロセスを改善する．

運用開始にあたっては，プロジェクトマネージャに対するセミナーを開催し，自分たちのプロジェクトのプロセスを定義する実習を行うことにより，理解を深めたのちに運用を始めた．

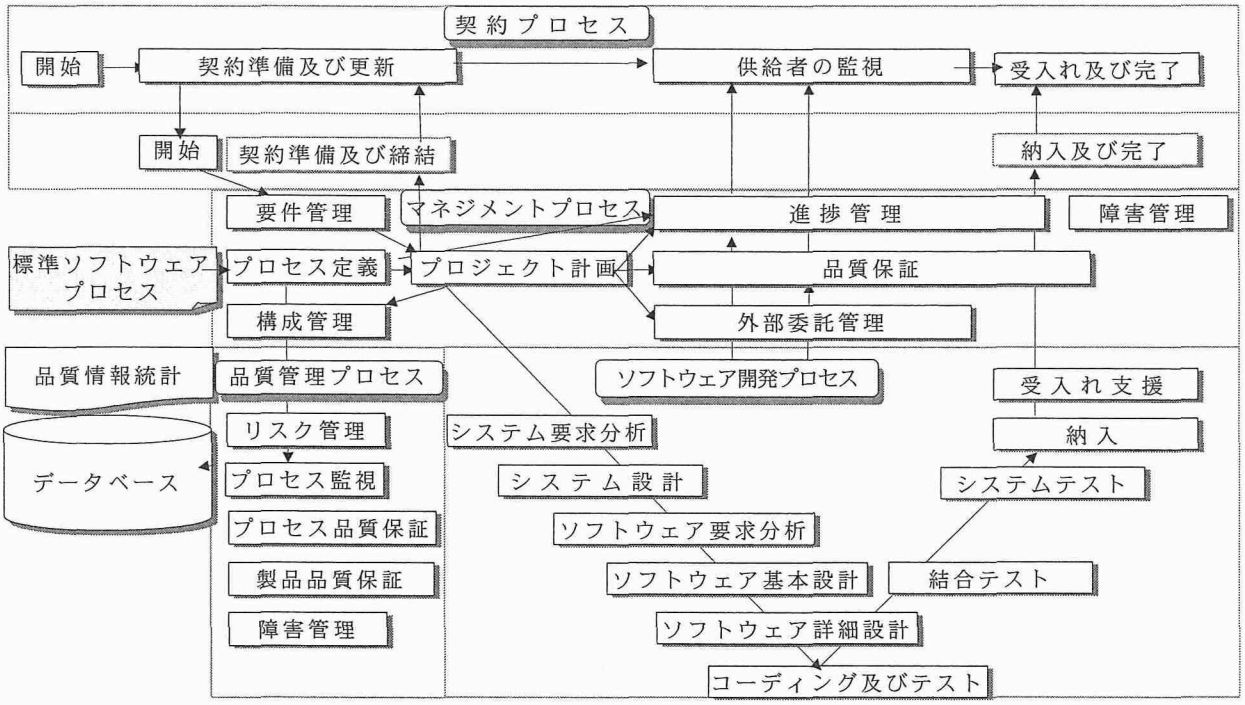


図 3.13 標準ソフトウェアプロセスの概要

表 3.3 マネジメントプロセス定義の実例(前半)

	INPUT 資料名称	活動項目	OUTPUT 成果物	判定基準
要件管理	要求仕様書 開発スケジュール 受入れ方針 見積り依頼書 見積書 見積り仕様書	○完成イメージの顧客との合意のために，また，プロジェクトの管理及び品質保証の枠組みのために， ・取得要求事項をレビューする． ・取得要求事項の変更をレビューする．	レビュー記録 工番開始通知 工番登録票	・取得要件がレビューされている． ・取得要件が変更される際に計画が変更されている．
プロセス定義	標準ソフトウェアプロセス	○標準ソフトウェアプロセスを基に，プロジェクトプロセスを定義する． ○プロジェクトプロセスを使用したプロジェクト計画を作成する． ○プロジェクトプロセスに従って，プロジェクトを管理する．	○プロジェクトプロセス ・契約の視点 ・マネジメントの視点 ・開発の視点	・プロジェクトプロセスが作成され計画に使用されている． ・プロジェクトプロセスと一貫性のあるプロジェクト管理が実施されている．
計画	要求仕様書 開発スケジュール 受入れ方針 見積り依頼書 見積書 見積り仕様書	○開発規模の計画，開発の進め方の計画，進捗管理ができる，事後評価できるために以下をおこなう． ・作業成果物を明確にし規模をレビューする． ・規模と見積り書を関連づける． ・スケジュールと規模と費用工数を関連づける． ・スケジュールをレビューする．	○プロジェクト計画書 ・プロジェクト体制（役割と責任の定義） ・環境，ツール，標準等エンジニアリング環境 ・作業項目構成，人員配置，作業成果物の規模 ・スケジュール ・共同レビュー計画 ・プロジェクトのプロセス定義 ○工数と費用の見積り	・開発計画の作成活動がされている． ・技術見積り，見積り前提条件，スケジュールがレビューされている． ・開発計画が成功シナリオと評価できる． ○以下は，スケジュールの判定条件 ・見積りとスケジュールの整合がとれている． ・スケジュール作業項目は詳細である． ・スケジュール作業項目毎に成果物が定義されていて，100%達成を確認できる． ・フェーズ毎の工数配分比に問題はない． ・マンパワー曲線の困難度は高くない． ・メンバースキルを把握している． ・不確定要素が抽出されている． ・過去の失敗原因を改善している．
		○品質保証計画をレビューする．	○品質保証計画書 ・レビューのスケジュール ・品質特性の管理	・品質保証計画がレビューされている．
		○リスクを分析し，優先順位をきめ対応策をたてる．	○リスク計算書 ・リスクの識別と評価	・リスクの識別と評価がレビューされている．
		○外部委託計画をレビューする．	外部委託管理計画	・外部委託管理計画がレビューされている．

表 3.3 マネジメントプロセス定義の実例（後半）

進捗管理	スケジュール プロジェクトのプロセス定義 品質保証計画書	○プロジェクトが今後どのように推移していくのか，QCDのゴールはどこなのかを予測するために， ・スケジュールをモニターする． ・作業成果物の規模をモニターする． ・工数と費用をモニターする． ・リスクをモニターしレビューする． ・計画と実績のギャップを分析する． ・計画が大きくかわれば再計画しレビューする．	プロジェクト計画書のモニター EV分析グラフ（計画工数，実際工数，達成価値⇒工数差異，時間差異，工数期待値）	・計画に対する進捗のモニターが実施されている． （費用，スケジュール，リスク，技術制約，性能） ・計画の改定活動が実施され，内容がレビューされている．
品質保証	品質保証計画書 作業成果物（計画，進捗，仕様書，設計書，テスト項目，テスト成績書）	○要求品質を顧客に保証するために，保証できるしくみを持ち，その仕組みで品質特性を検証する． ・品質保証計画書をレビューする． ・品質保証計画書にそって活動する． ・開発計画をレビューする． ・開発活動をレビューする． ・作業成果物をレビューし，不適合事項を文書化しモニターする． ・顧客指定の標準，手順，要件と照合し，評価する． ・出荷の準備において，製品が顧客の要求事項を完全に満たし，かつ取得者が受入れ可能であることを保証する．	○品質保証計画書 ・参画するレビュー ・実施する診断 ・納入対象成果物のレビュー レビュー記録 設計品質計測シート 品質結果報告書 工番完了票	・品質を判断する客観的な基準をもちいている． ・プロセスの進め方から，開発対象の品質を予測している． ・品質保証活動が実施され報告されている． ・設計品質計測シート，品質指標，信頼性評価がレビューされている．
構成管理		○ソフトウェアライフサイクルを通じてソフトウェア成果物を管理するために，構成管理計画を立案する． ○文書体系規程にしたがい，体系を確立する． ○構成制御（変更管理）し，状況及び履歴を残す．（バージョン，リリース識別，リリース間の比較） ○構成品目の完全性を保証する． ○リリース管理をおこなう．	（構成管理計画） （管理記録） （ﾘﾘｰｽ管理）	・構成管理計画がレビューされている． ・管理記録が報告されている．
障害管理	クレーム速報	○受け入れ，運用保守のステージで発見された問題を分析し解決する． ○構成管理のもとリリースする． ○深層原因を求め，その傾向を把握し，標準プロセス定義を改善する．	クレーム速報 クレーム報告書	・即時対応されている． ・対応が顧客満足を得られている． ・構成管理されている． ・深層原因が究明され適切な再発防止策で悪化傾向が改善されている．

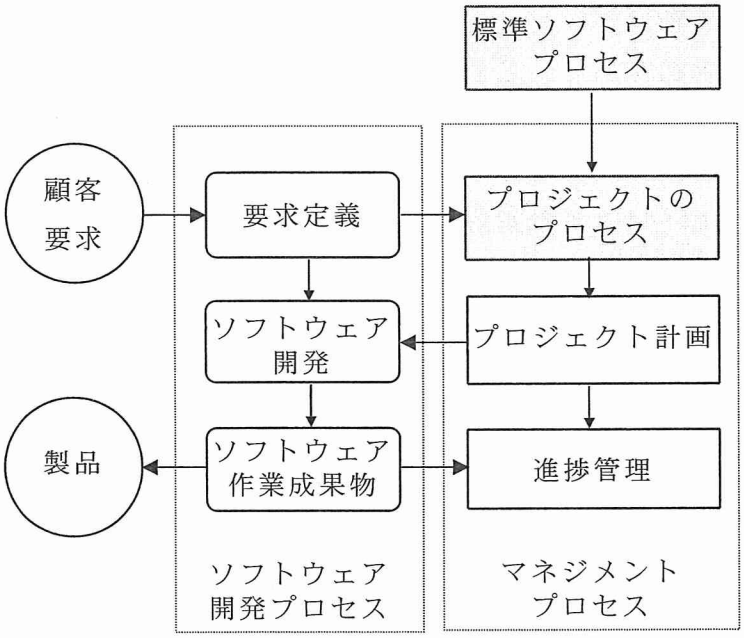


図 3.14 標準ソフトウェアプロセスの使い方

3.3.2 品質管理プロセスの事例

品質管理部門の役割は，ソフトウェア開発プロジェクトが QCD に関するすべての指標において成功することを目的に，プロジェクトの状況を測定し，目標達成の支援を行い，ソフトウェア製品品質を保証することである．図 3.15 に，構築した品質管理プロセスとソフトウェア開発プロジェクトとの係わりを示す．リスク管理活動は，プロジェクトのリスクを識別し，それを評価し，リスク管理計画を展開し，追跡する活動である．プロセス品質保証活動は，プロジェクト計画のレビューとプロジェクト進捗のモニタで，プロジェクトマネジメントを支援し管理する活動である．製品品質保証活動は，レビューとテスト結果の評価により，ソフトウェア製品の品質を保証する活動である．これらの活動によって，リスクが問題として発生していないかどうかを確認し，プロジェクトの失敗を未然に防止している．表 3.4 に品質管理プロセス定義の実例を示す．

リスク管理活動，プロセス品質保証活動，製品品質保証活動は，通常それぞれの専任者が担当する．このため，各専任者はプロジェクトの情報を共有し，コミュニケーションを密にすることが重要である．専任者の連携により，リスク管理活動はリスクが制御されているか事実確認ができる．プロセス品質保証活動と製品品質保証活動は，注力すべきプロジェクトがどれかプロジェクトの優先順位を知ることができる．この仕組みにより，ハイリスクプロジェクトの見逃しを無くすることができプロジェクトを成功に導くことができる．

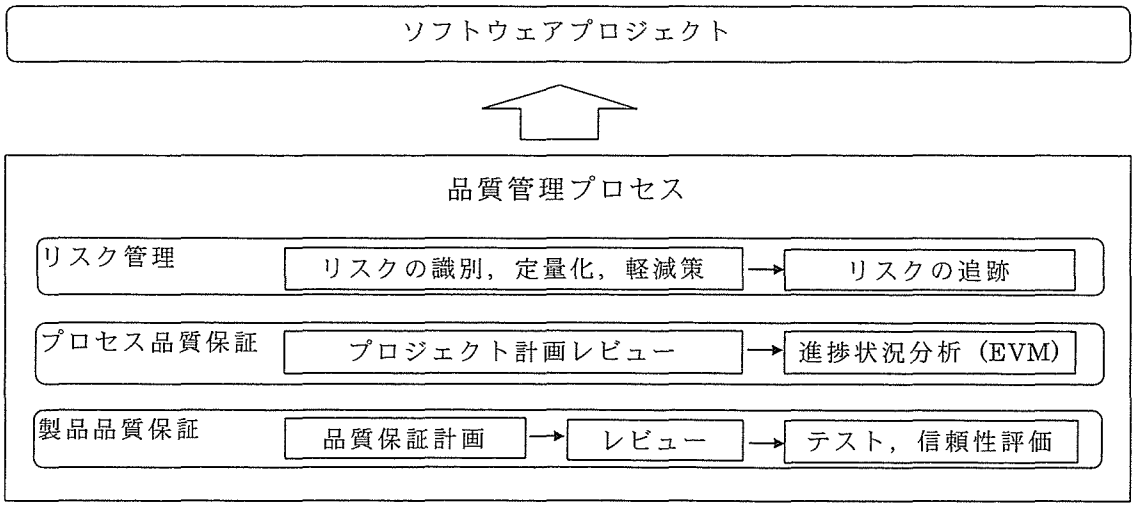


図 3.15 品質管理プロセスの概要

表 3.4 品質管理プロセス定義の実例

	INPUT 資料名称	活動項目	OUTPUT 成果物	判定基準
リスク管理	プロジェクト登録票 リスク計算書 プロジェクト計画・実績 EVM分析 品質保証計画・実績 設計品質計測シート プロジェクト完了票	○要件管理・プロジェクト計画から開始時リスクを早期発見するために，および品質保証から開発時リスクを早期発見するために， ・技術PMと品質保証を対象にリスク管理定例会を開催する． ・リスク要因を分析し，リスク度を算出し，対象プロジェクトのリスク順位評価する． ・リスク軽減策の計画を支援し，進捗をフォローする． ・プロセス品質保証のレビューを行い，その結果を報告する． ・製品品質保証のレビューを行い，その結果を報告する．	定例会議事録 プロジェクトフォローシート リスク管理順位 リスク管理履歴	・定例会が開催されている． ・リスク順位が評価されている． ・リスク軽減策の計画と進捗がフォローされている．
プロセス管理	プロジェクトのプロセス	○標準ソフトウェアプロセスを基に，プロジェクトプロセスが定義されているかモニターする． ○プロジェクトプロセスを使用したプロジェクト計画が作成されているかモニターする． ○プロジェクトプロセスと一貫性あるプロジェクト管理が行なわれているかモニターする． ○標準ソフトウェアプロセスに問題ある場合，標準を改訂する．	モニター結果 改定した標準ソフトウェアプロセス	・プロジェクトプロセスの定義活動と運用をモニターしている． ・必要がある場合に標準ソフトウェアプロセスが改定されている．
プロセス品質保証	レビュー記録 プロジェクト計画，実績	○品質を保証できる仕組みで，プロセスが計画どおり実施されていることを保証する． ・要件管理活動とその作業成果物のレビューを行い，その結果を報告する． ・プロジェクト計画のレビューを行い，その結果を報告する． ・進捗管理活動のレビューを行い，その結果を報告する． ・検証／妥当性確認活動のレビューを行い，その結果を報告する． ・外部委託管理活動とその作業成果物のレビューを行い，その結果を報告する． ○活動の成果物は顧客が利用できるようにする．	品質保証計画 プロジェクト計画 チェックシート EV分析を用いた予測値 プロジェクト完了票	・要件の変更活動をレビューしている． ・プロジェクト計画の妥当性評価が報告されている． ・プロセスの進め方から，開発対象のQCDのゴールを予測している． ・品質を判断する客観的な基準をもちいている．
製品品質保証	品質保証計画書 作業成果物（仕様書，設計書，テスト項目，テスト成績書）	○製品が，設定した要求事項に適合していることを保証する． ・品質保証計画書を作成し文書化しレビューし，再計画する． ・品質保証計画書にそって活動する． ・作業成果物をレビューし，不適合事項を文書化しモニターする． ・システムテスト結果から信頼性評価する． ・妥当性確認結果から信頼性評価する． ・顧客指定の標準，手順，要件と照合し，評価する． ○活動の成果物は顧客が利用できるようにする． ○プロジェクトが検証／妥当性確認を必要としているのか決定する．	品質保証計画書 設計品質計測シート 信頼性評価 プロジェクト完了票	・品質を判断する客観的な基準を用いている． ・品質保証活動が実施され報告されている． ・設計品質計測シート，品質指標，信頼性評価がレビューされている．
検証／妥当性確認	要求仕様書	○要求事項を満たした製品が，明確に意図した使用方法を満足していることを確認する． ・必要な場合，妥当性確認計画を作成し文書化しレビューする． ・妥当性確認のテストケース（品質指標）を作成しレビューする． ・テストを実施する． ・妥当性確認結果をレビューし評価する． ○活動の成果物は顧客が利用できるようにする．	検証計画 品質指標 検証結果報告書	・品質を判断する客観的な基準をもちいている． ・品質指標がレビューされている． ・妥当性確認結果がレビューし評価されている．

3.4 おわりに

本章では，ソフトウェア開発における QCD の目標を実現するために必要なソフトウェアプロセスの改善技術として，ソフトウェアプロセスの考え方，ソフトウェアプロセスモデル CMM をガイドラインとしたプロセスの改善方法，さらに改善したソフトウェアプロセスをソフトウェア開発が成功するためのシナリオとして，運用していくための標準ソフトウェアプロセスについて議論した．

現実に，ソフトウェア開発に携わるプロジェクトマネージャは，「品質が悪い，納期に間に合わない，予算がオーバーしそう」などの悩みをもっている場合が多い．このプロジェクトが成功するかどうかの不安は，ソフトウェア開発の最初の計画段階からもっているものである．このような悩みを軽減しプロジェクトの目標に立ち向かうために，ソフトウェアプロセスの改善技術が有効である．プロセスの改善技術を有効に活用することにより，プロジェクトで発生するプロセスの課題から自分たちのソフトウェアプロセスを改善し，ソフトウェア技術者のマネジメント技術を向上する PDCA の管理のサイクルを持続して繰り返して行くことができる．

プロセス改善とは，ソフトウェア開発を成功させる仕組みを求め続けることである．ソフトウェア開発を成功させる仕組みを開発し保守するために，プロセス改善技術は，重要な技術であると考える．

第4章 ソフトウェアプロジェクトの定量的評価法

ソフトウェア製品品質の向上を図るために，プロジェクト開始の早い段階でその品質を予測しソフトウェア開発プロセスに対して適切な対策を実施するプロジェクトマネジメント技術が重要である．プロジェクトマネジメント技術は，ソフトウェア開発プロジェクトから収集された計測データを統計的に分析し，データに基づいた改善のPDCAサイクルの確立を実現し，継続して向上していくことが重要である．その繰り返しの中から，ソフトウェア開発の予測性，制御性，効率性をより向上させる方法を導き出していくことにより，プロジェクトを成功に導くことができる．

本章では，実際のプロジェクトの測定データを用いてプロジェクトの定量的評価を行い，ソフトウェア製品品質および開発コストの目標値を達成するためのプロジェクトマネジメント技術の有効性について議論する．

4.1 はじめに

3.3 にて示したように，プロジェクトマネージャは，マネジメントプロセス定義（表 3.3 参照）に基づいて，プロジェクト計画や進捗管理などのプロジェクトマネジメントを実施している．品質管理部門は，品質管理プロセス定義（表 3.4 参照）に基づいて，リスク管理，プロセス品質保証，および製品品質保証などの活動を実施している．

本章では，これらの活動をソフトウェアプロジェクトの QCD 結果により定量的に評価するために，これらの活動で収集された計測データを用いた重回帰分析を行い，プロジェクト開始の早い段階でソフトウェアプロジェクトの QCD 結果を定量的に予測することが可能な予測モデル式を導出する．

予測モデル式の考察から，プロジェクトマネジメントや品質保証活動がソフトウェアプロジェクトの QCD 結果に及ぼす様々な要因の影響を検討し，より効果的なマネジメントプロセスについて分析する．

ここで，多変量解析の手法である重回帰分析は，目的変量（Y）と説明変量（X_i）の間の因果関係を，線形式により調べるものである．重回帰分析を行うと，結果といくつかの原因を結びつける関係式がわかる．その関係式から，結果に大きな影響を与えている原因を明らかにでき，結果を予測し，結果を良くするための要因を改善することができると言われている．

4.2 マネジメント要因による QCD 予測

4.2.1 予測モデル式の導出

プロジェクトの結果は，リスク管理，プロジェクト管理，品質保証などのマネジメント活動に依存すると考える．そこで，これらの活動とプロジェクト結果の因果関係を導き出すために，これらの活動によって収集されたデータを用いて重回帰分析を行い，下記の予測モデル式を導出する．予測モデル式に基づいて，顧客納入後の製品品質と開発コストに影響を及ぼす要因とその影響度を分析する．

ソフトウェア製品品質
　＝F(リスク管理活動，プロジェクト管理活動，品質保証活動)，　(4.1)
ソフトウェア開発コスト
　＝G(リスク管理活動，プロジェクト管理活動，品質保証活動)．　(4.2)

ここで，式(4.1)の F( )および式(4.2)の G( )は重回帰式である．

目的変量としては，ソフトウェア製品品質を測るメトリクスは納入後フォールト数(*Y1*)を，ソフトウェア開発コストを測るメトリクスはコスト予実差(*Y2*)を用いる．

説明変量としては，リスク管理活動の影響因子は開始時リスク度(*X1*)とリスク度軽減速度(*X2*)を，プロジェクト管理活動の影響因子は規模当り EVM 回数(*X3*)を，製品品質保証活動の影響因子は規模当りレビュー回数(*X4*)とレビュー合格率(*X5*)を用いる．

各諸量を以下に説明する．

Y1：納入後フォールト数＝受入れ試験フォールト数とフィールド障害数との合計値．

Y2：コスト予実差＝コスト実績値／ソフトウェア開発予算．

X1：開始時リスク度＝Σ_i(識別したリスク項目×項目毎配点)$_i$（2.2.8 参照）．

X2：リスク度軽減速度＝リスク度が 30 点以下に到達するまでの期間／開発予定期間（2.2.8 参照）．

X3：規模当り EVM 回数=単位開発規模当りの EV 分析回数
EVM の実施が頻繁なプロジェクトでは，プロジェクトマネージャが早期に問題に対応することにより，早期にリスクを軽減できている経験による．

X4：規模当りレビュー回数=単位開発規模当りのレビュー実施回数
高い品質意識をもつプロジェクトでは，頻繁にレビューを実施している経験による．

X5：レビュー合格率＝レビュー初回合格率.
レビュー合格率でレビュアの判定レベルを測る.

4.2.2 分析データ

予測モデル式のパラメータを推定するために用いたデータは,実際の 10 個の異なるソフトウェア開発プロジェクトから得られた表 4.1 のデータである．予測モデル式における５つの説明変量 *X1*，*X2*，*X3*，*X4*，および *X5* と，目的変量 *Y1* および *Y2* を，表 4.1 のデータを用いて相関分析を行うと表 4.2 が得られる.

表 4.2 から下記の相関関係が認められる.

- 納入後フォールト数(*Y1*) およびコスト予実差(*Y2*)は，開始時リスク度(*X1*)，およびリスク度軽減速度(*X2*)と正の相関があり，重回帰分析が可能である.
- 開始時リスク度(*X1*)は，リスク度軽速度(*X2*)および規模当り EVM 回数(*X3*)との相関も高い.
- 納入後フォールト数(*Y1*)は，レビュー合格率(*X5*)との相関は見られない.

4.2.3 ソフトウェア製品品質の重回帰分析

表 4.1 のデータを用いて納入後フォールト数(*Y1*)を目的変量とする複数の重回帰式を求め，求めた重回帰式の適合性を決定係数 R^2，補正決定係数 R^2 で確認することにより，予測に有効な説明変量としてリスク度軽減速度(*X2*)，規模当り EVM 回数(*X3*)，規模当りレビュー回数(*X4*)の３要因を選択した．この３つの説明変量による重回帰分析の分析結果を，表 4.3～表 4.5 に示す．推定された重回帰式を式(4.3)に示す．また，分析するデータを標準化して重回帰分析を実施すると，標準化重回帰式として式(4.4)を得た.

$$Y1 = 7.6944X2 - 0.7582X3 - 3.3709X4 + 1.4190, \tag{4.3}$$

$$\text{標準化 } Y1 = 0.6426X2 - 0.1181X3 - 0.3301X4. \tag{4.4}$$

式(4.3)により，説明変量の目的変量に影響を与える度合いは，*X3* ＜ *X4* ＜ *X2* である．リスク度軽速度(*X2*)が，顧客納入後のソフトウェア製品フォールト数に大きな影響を与えている.

式(4.3)による予測値の精度を図 4.1 に示す．推定された決定係数が示すように，図 4.1 は実績値と予測値との間の相関性は低いことを意味しており，品質予測式としての式(4.3)の予測精度は余り高くない．精度を高くできない原因として，目的変量である顧客納入後のソフトウェア製品フォールト数が６プロジェクトにおいて 0

であった影響を受けていると考えられる．また，今回対象とした説明変量以外にも重要な要因はあると思われる．今回の 10 個のソフトウェア開発プロジェクトは，全て異なるプロジェクトチームであった．複数組織におけるソフトウェア製品品質を予測するには，組織特性要因の影響も検討すべきであると考える．

しかし，実際に納入後にフォールトが多く発生しているプロジェクトに対してはフォールトが多いと予測することができている．このことから，ソフトウェア製品品質を評価するために，品質予測式は有効であると判断する．品質予測式を用いると，開発の初期におけるリスク管理，プロジェクト管理，およびレビューの計画がソフトウェア製品品質に適切に有効であったかどうかを確認することができる．

4.2.4 ソフトウェア開発コストの重回帰分析

表 4.1 のデータを用いて，コスト予実差(*Y2*)を目的変量とする複数の重回帰式を求め，求めた重回帰式の適合性を決定係数 R^2，補正決定係数 R^2 で確認する．これにより，予測に有効な説明変量として，リスク度軽減速度(*X2*)，規模当り EVM 回数(*X3*)，規模当りレビュー回数(*X4*)の 3 要因を選択した．この 3 つの説明変量による重回帰分析の分析結果を，表 4.6～表 4.8 に示す．推定された重回帰式を式(4.5)に示す．また，分析するデータを標準化して重回帰分析を実施すると，標準化重回帰式として式(4.6)を得た．

$$Y2 = 0.1607\,X2 - 0.0314\,X3 + 0.1077\,X4 + 0.9307, \tag{4.5}$$

$$\text{標準化 } Y2 = 0.6012\,X2 - 0.2189\,X3 + 0.4726\,X4. \tag{4.6}$$

式(4.6)により，説明変量の目的変量に影響を与える度合いは，$X3 < X4 < X2$ である．リスク度軽減速度(*X2*)が，ソフトウェア開発コストにも大きな影響を与えている．

式(4.5)による予測値と実測値を図 4.2 に示す．決定係数が示すように，図 4.2 はソフトウェア開発コストを高い精度で予測できることを示している．プロジェクトの早い時期にリスク軽減策に手を打つことが出来れば，コスト管理も適切に行えると考える．

4.2.5 マネジメント要因の有効性評価

（1） 予測モデル式によるリスク管理の評価

我々の経験プロジェクトにおいて実施しているマネジメントのうち，リスク管理，プロジェクト管理，品質保証の 3 つを比較すると，リスク管理がプロジェクトの結

果に最も有効なマネジメントであるという評価結果となった．これは，３つのマネジメントの中では，リスク管理だけがプロジェクトマネージャを直接支援する活動であるためと考える．

実際のプロジェクトにおいては，プロジェクトマネージャのマネジメントスキルの低さに起因して，QCD の失敗を起こすことが多い．マネジメントスキルが低いプロジェクトマネージャは，プロジェクトの計画精度が低くスケジュールの予測も貧弱である．リスク管理は，このマネジメントリスクを軽減するために，プロジェクトマネージャのマネジメント技術を支援する活動である．

予測モデルの重回帰分析により，リスク管理によるプロジェクトマネージャへの支援活動が，プロジェクトの成功に有効であることを確認できた．今後もより一層プロジェクトマネージャを支援していかなければならない．

なお，リスク管理の有効性評価を下げる原因について調査すると，プロジェクトチームが顧客の会社に常駐することによりプロジェクトマネージャを支援できなくなった場合や，顧客担当分のハードウェア遅れなどから影響を被った場合であった．

（２） 予測モデル式による製品品質保証の評価

品質保証担当者のレビュー評価は，製品品質に対して有効に機能していないという評価結果となった．このことから，レビューの評価方法に問題があるものと考えられる．

品質保証担当者がレビューに参加して品質を評価する目的は，設計レビューにおいては仕様書記載事項の欠落がないか，仕様書との関係をすべて読み取れるかなど，開発工程の各マイルストーンで品質を確保するためである．有効に機能していない原因を調査し，品質保証担当者のレビュー評価の甘さを検出することができた．以後は，リスク管理によりレビュー評価の内容を確認し，同時にレビュー技術の支援も行うようにしている．

表 4.1　分析データ

	リスク管理活動		プロジェクト管理活動	製品品質保証活動		ソフトウェア製品品質	ソフトウェア開発コスト
プロジェクト No.	開始時リスク度 *X1* (0～100)	リスク度軽減速度 *X2* (0～1.00)	規模当り EVM回数 *X3* (回／規模)	規模当り レビュー回数 *X4* (回／規模)	レビュー合格率 *X5* (0～1.00)	納入後フォールト数 *Y* (件)	コスト予実差 *Y2* (差無=1)
1	73	1.00	0.21	0.05	0.50	12	1.13
2	47	0.33	0.78	0.10	0.50	0	0.92
3	38	0.87	1.18	0.14	0.50	13	1.12
4	35	0.95	0.74	0.12	1.00	3	1.04
5	34	0.51	2.70	0.22	0.50	0	0.93
6	32	1.00	1.70	1.71	0.67	1	1.19
7	32	0.28	0.79	0.07	0.00	0	0.92
8	24	0.00	0.99	0.62	0.74	0	1.03
9	23	0.00	1.84	0.31	0.50	0	0.82
10	13	0.00	2.42	0.54	1.00	0	1.00

表 4.2 相関分析表

	X1	*X2*	*X3*	*X4*	*X5*	*Y1*	*Y2*
X1	1						
X2	0.6328	1					
X3	-0.6563	-0.3538	1				
X4	-0.3376	0.1248	0.3321	1			
X5	-0.3018	0.0495	0.2377	0.2903	1		
Y1	0.6507	0.6432	-0.4551	-0.2891	-0.0722	1	
Y2	0.3517	0.7377	-0.2746	0.4750	0.3019	0.5889	1

表 4.3　回帰精度

重相関係数 R	0.7500
決定係数 R^2	0.5625
補正決定係数 R^2	0.3438
標準誤差	4.1737

表 4.4 分散分析表

	自由度	変動	分散	検定統計量F_0	有意 F
回帰	3	134.4	44.79	2.5715	0.1498
残差	6	104.5	17.42		
合計	9	238.9			

表 4.5 推定された回帰係数

	偏回帰係数	標準誤差	検定統計量 t	標準偏回帰係数	標準誤差
切片	1.4190	3.7329	0.3801	0.0000	0.2700
X2	7.6944	3.5950	2.1403	0.6426	0.3003
X3	-0.7582	2.0276	-0.3739	-0.1181	0.3158
X4	-3.3709	3.0405	-1.1086	-0.3301	0.2977

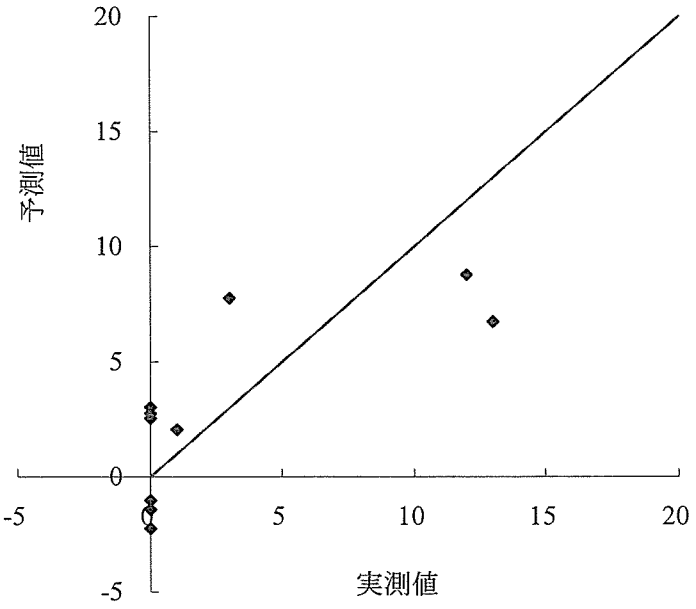


図 4.1　顧客納入後フォールト数の予測精度

表 4.6　回帰精度

重相関係数 R	0.8533
決定係数 R^2	0.7281
補正決定係数 R^2	0.5922
標準誤差	0.0734

表 4.7 分散分析表

	自由度	変動	分散	検定統計量F_0	有意 F
回帰	3	0.0866	0.0289	5.3568*	0.0392
残差	6	0.0324	0.0054		
合計	9	0.1190			

(*：危険率5%で有意)

表 4.8 推定された回帰係数

	偏回帰係数	標準誤差	検定統計量 t	標準偏回帰係数	標準誤差
切片	0.9307	0.0657	14.1711	0.0000	0.2700
X2	0.1607	0.0632	2.5402	0.6012	0.2367
X3	-0.0314	0.0357	-0.8791	-0.2189	0.2490
X4	0.1077	0.0535	2.0139	0.4726	0.2347

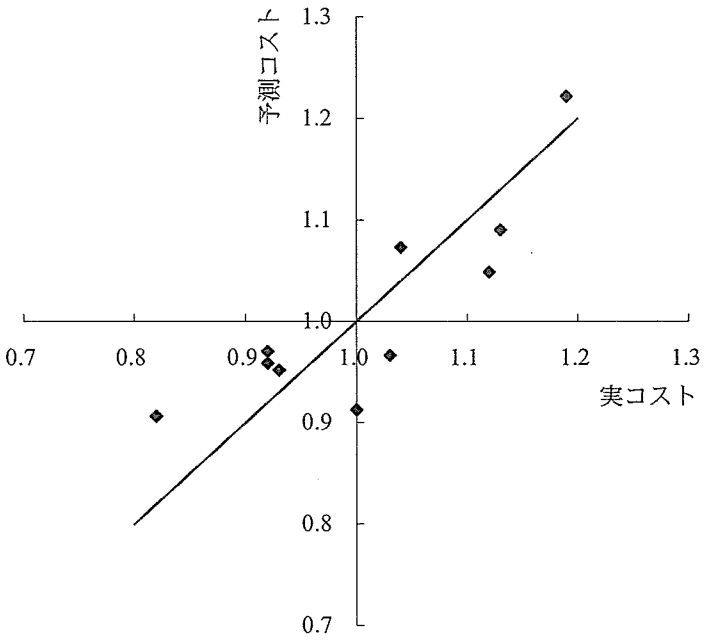


図 4.2 ソフトウェア開発コストの予測精度

4.3 品質保証要因による品質予測

4.3.1 予測モデル式の導出

ソフトウェア開発ライフサイクルにおけるフォールトの潜入と，レビューあるいはテストによるフォールトの除去推移の傾向変化の理想と実際を図 4.3 で示す．ソフトウェア開発において，要求定義や設計，製造プロセスで作り込まれるフォールトは，レビューによって早期に除去され，テストプロセスで全てのフォールトが除去された後出荷されるのが理想である[14],[15]．しかし実際には，レビュー進捗の遅れにより図 4.3 のフォールトの検出曲線が右方向にずれたり，あるいはレビュアによるフォールトの除去が少ないためフォールトの検出曲線は上へ押し上げられたりする．また，不十分なテスト項目数によってもテストプロセスでフォールトを完全に除去できない事態に陥りやすい．これらの結果，予定完了日には多くのフォールトが残り，出荷延期や，あるいは顧客納入後のフォールトに対応しなければならなくなる．

よって，ソフトウェア製品の品質は，レビューのマネジメント，レビュアの評価技術，およびテスト項目抽出技術の影響を受けている．そこで，これらの要因とソフトウェア製品品質の関係を導き出すために，下記の品質予測モデル式を推測し重回帰分析を行う．予測モデル式に基づいて，顧客納入後の製品品質に影響を及ぼす要因とその影響度を分析し，より効果的なマネジメントを議論する．

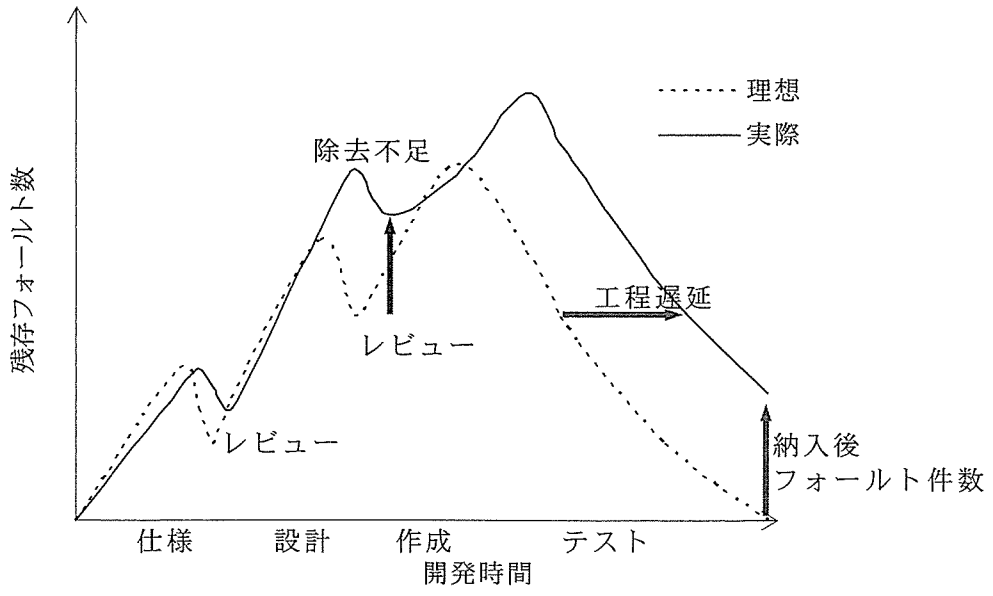


図 4.3　フォールトの作り込みと除去の推移の傾向変化

ソフトウェア製品品質

$=F$(レビューマネジメント，レビュー評価技術，テスト項目抽出技術). (4.7)

ここで，式(4.7)の $F(\ )$は重回帰式である．

目的変量として，ソフトウェア製品品質を測るメトリクスは，総合検証検出フォールト数(*Y*)を用いる．説明変量として，レビューマネジメントの影響因子は設計レビュー遅れ率(*X1*)と規模当りレビュー回数(*X2*)を，レビュー評価技術の影響因子はレビュー評価平均点(*X3*)を，テスト項目抽出技術の影響因子は総合検証項目数(*X4*)を用いる．

各諸量を以下に説明する．

Y ：総合検証検出フォールト数＝システムテスト検出フォールト数と品質保証部門の検査検出フォールト数．

X1：設計レビュー遅れ率＝設計レビュー遅れ期間平均／開発予定期間．
レビューが予定より進んでいる場合はマイナスとなる．

X2：規模当りレビュー回数＝単位開発規模当りのレビュー回数．

X3：レビュー評価平均点＝設計レビューの評価平均点．
レビューの評価点は，レビューチェックリストにより測定される．100 点満点である．

X4：総合検証項目数＝システムテスト実施項目数と品質保証部門の検査実施項目数．

4.3.2 分析データ

ソフトウェア製品品質予測モデル式のパラメータを推定するために用いたデータは，1 開発部門における 1 システムに対する 7 個のソフトウェア派生開発から得られた表 4.9 のデータである．1 つの開発部門に絞るのは，組織特性要因を排除するためであり，また目的変量として 0 を含みにくい総合検証検出フォールト数を選択した（4.2 参照）．

予測モデル式における 4 つの説明変量 *X1*, *X2*, *X3*, および *X4* と，目的変量 *Y* を，表 4.9 のデータを用いて相関分析を行うと表 4.10 が得られる．

表 4.10 から下記の相関関係が認められる．

- 総合検証検出フォールト数(*Y*)は，設計レビュー遅れ率(*X1*)と正の相関が，規模当りレビュー回数(*X2*)と負の相関があり，重回帰分析が可能である．
- 総合検証検出フォールト数(*Y*)はレビュー評価平均点(*X3*)との相関は低い．

・規模当りレビュー回数(*X2*)とレビュー評価平均点(*X3*)の相関も高い．

4.3.3 重回帰分析

表 4.9 のデータを用いて総合検証検出フォールト数(*Y*)を目的変量とする複数の重回帰式を求め，決定係数 R^2 と補正決定係数 R^2 を比較することにより，予測に有効な説明変量として，設計レビュー遅れ率(*X1*)，規模当りレビュー回数(*X2*)，総合検証項目数(*X4*)の 3 要因を選択した．この説明変量による重回帰分析の分析結果を表 4.11～表 4.13 に示す．推定された重回帰式を式(4.8)に示す．また，分析するデータを標準化して重回帰分析を実施すると，標準化重回帰式として式(4.9)を得た．

$$Y=41.7178X1-1.5020X2+0.0644X4+18.7120, \tag{4.8}$$

$$\text{標準化 } Y=0.6126X1-0.5413X2+0.1958X4. \tag{4.9}$$

この重回帰式は，表 4.12 の分散分析表により，予測に役立たないという帰無仮説は棄却される．式(4.9)により，説明変量の目的変量に影響を与える度合いは，*X4* ＜ *X2* ＜ *X1* である．設計レビュー遅れ率(*X1*)と規模当りレビュー回数(*X2*)が総合検証検出フォールト数に大きな影響を与えている．

式(4.8)による総合検証検出フォールト数の予測精度を図 4.4 に示す．決定係数が示すように，図 4.4 は総合検証フォールト数を高い精度で予測できることを示している．

4.3.4 品質保証要因の有効性評価

（1） 予測モデル式によるレビューマネジメントの評価

我々の経験プロジェクトにおいて実施している品質保証活動のうち，レビューのマネジメント，レビュアの評価技術，テスト項目の抽出技術の 3 つを比較すると，レビューのマネジメントがソフトウェア製品品質に最も有効であるという評価結果となった．これは 3 つの品質保証活動の中では，レビューのマネジメントだけがプロセス品質を扱うマネジメントであるためと考える．

実際のプロジェクトには，プロジェクトマネジメントが貧弱な場合にレビューの進捗が遅れることが多い．進捗が遅れた後にレビューを省略する事態に陥ることもある．プロジェクトがこのような状況に陥った場合は，QCD 目標を達成できない結果となるものである．このような経験から，リスク管理活動におけるリスクの追跡と監視の判断基準においても，レビューの進捗管理を用いている（2.2.6 参照）．すなわち，計画に対するレビュー実施遅れは 10%以内か，全てのレビューに合格して

いるかを判断基準としている.

予測モデルの重回帰分析により，プロセス品質を保証するレビューの計画と進捗管理が，ソフトウェア製品品質の向上に有効であることを確認できたと考える.

（2） 予測モデル式によるレビュア評価技術の評価

マネジメント要因による分析結果と同じく，品質保証担当者のレビュー評価は製品品質に対して有効に機能していないという評価結果が得られた．前述したリスク管理によるレビュー評価の対策実施において，実際にその後のプロジェクトの追跡調査をした結果を 4.3.5 にて示す.

表 4.9　分析データ

	レビューマネジメント		レビュア 評価技術	テスト項目 抽出技術	ソフトウェア 製品品質
プロジェクト No.	設計レビュー 遅れ率 *X1* (-1.0～1.0)	規模当り レビュー回数 *X2* (回)	レビュー評価 平均点 *X3* (0～100)	総合検証 項目数 *X4* (項目)	総合検証検出 フォールト数 *Y* (件)
1	0.50	5	80.50	71	38
2	-0.07	12	82.50	48	3
3	-0.12	4	64.00	12	7
4	0.10	1	35.50	44	20
5	0.09	3	62.00	138	26
6	0.08	14	89.25	25	0
7	0.00	3	44.25	33	22

表 4.10　相関分析表

	X1	*X2*	*X3*	*X4*	*Y*
X1	1				
X2	-0.1343	1			
X3	0.2088	0.8384	1		
X4	0.3694	-0.2725	-0.0041	1	
Y	0.7576	-0.6769	-0.3234	0.5695	1

表 4.11　回帰精度

重相関係数 R	0.9705
決定係数 R^2	0.9419
補正決定係数 R^2	0.8839
標準誤差	4.6962

表 4.12　分散分析表

	自由度	変動	分散	検定統計量F_0	有意 F
回帰	3	1073.6	357.85	16.2259*	0.0233
残差	3	66.2	22.05		
合計	6	1139.7			

(*：危険率5%で有意)

表 4.13　推定された回帰係数

	偏回帰係数	標準誤差	検定統計量 t	標準偏回帰係数	標準誤差
切片	18.7120	4.3145	4.3370	0.0000	0.1391
X1	41.7178	10.2015	4.0894	0.6126	0.1498
X2	-1.5020	0.4015	-3.7411	-0.5413	0.1447
X4	0.0644	0.0508	1.2689	0.1958	0.1543

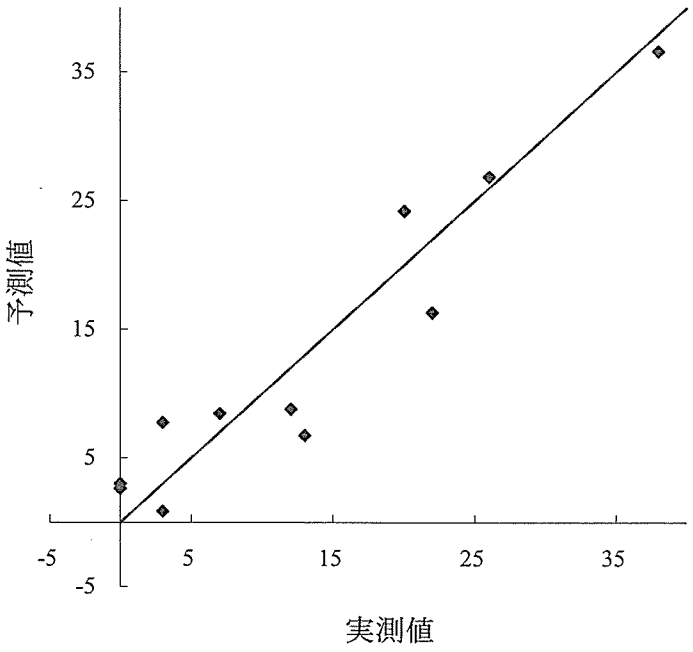


図 4.4　総合検証検出フォールト数の予測精度

4.3.5 その後の追跡と改善

（１） 品質予測式の追跡

品質予測式は本当に有効か，実際にその後のプロジェクトを式(4.8) で予測した結果，総合検証検出フォールト数は予測値 30.09 に対して実測値 39 となった．予測式はかなり有効であると評価できる．

（２） レビュー評価の追跡

設計レビューの評価方法の問題に関して，設計レビューによって指摘された欠点数，およびテストで検出したフォールトについて作り込み原因フェーズ毎の件数を計測した．実際のフォールトの作り込みと除去の推移を分析した結果を図 4.5 に示す．設計レビューで指摘された欠点数は，設計段階で作り込んだフォールト数に比べて極めて少なく，改善が必要と判明した．

（３） 弱点の改善

設計段階で作り込んだフォールトについて，レビューでフォールトを除去できなかった原因を分析し，レビュー評価基準の改善を図った．その後，開発部門と品質管理部門の共同レビュー不足の解消や，レビュー計画に対する進捗管理不足のマネジメントを改善した．

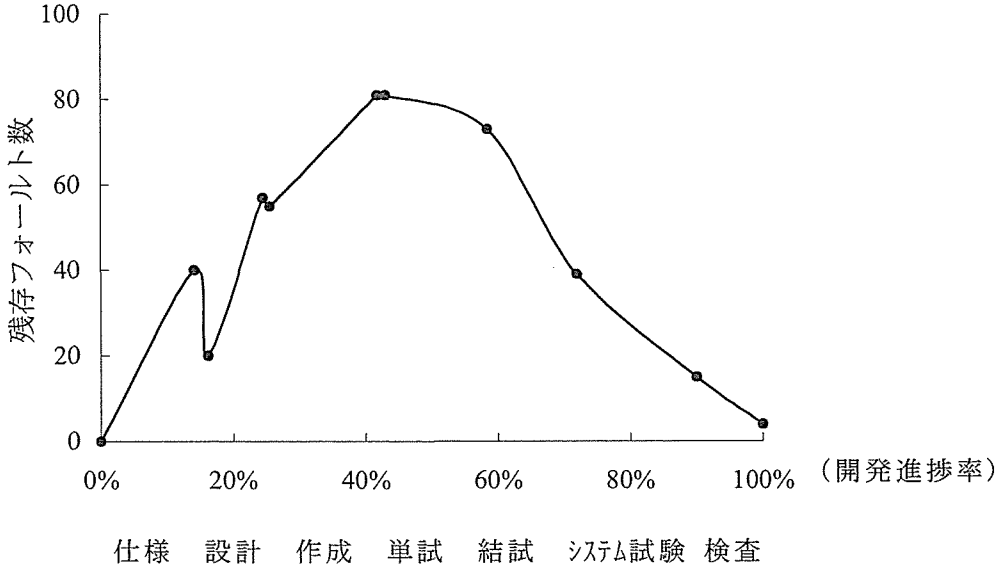


図 4.5 実際のフォールトの作り込みと除去の推移

4.4 おわりに

本章では，リスク管理，プロセス品質保証，および製品品質保証活動の測定データに着目し，これらの活動がソフトウェア製品品質や開発コストに及ぼす影響を分析した．重回帰分析により，これらの活動とソフトウェア製品品質の間には明瞭な相関関係があった．顧客納入後のソフトウェア製品品質およびソフトウェア開発コストの改善を図るには，マネジメント要因ではリスク管理活動により早期にリスクを軽減することが有効である．品質保証要因では，適切な回数のレビュー計画立案と計画通りに実施することが有効である．さらに開発プロセスの弱点として，レビュー評価方法の甘さが明らかになった．重回帰分析を用いたソフトウェアプロセスの定量的な評価は，プロセス改善を推進することに有効な手法であると考える．

課題としては，計測項目の少なさが上げられる．今後，レビューにおける指摘数を追跡調査した事例のように測定メトリクスの数を増やしていく必要があろう．ソフトウェア開発プロセスは，ソフトウェア製品品質に影響を及ぼす多くの要因がある．継続的な改善が重要である．

今後は，ソフトウェアプロジェクトの定量的評価を，すべてのソフトウェアプロジェクトに適用していくことにより，定量的なプロセスデータを計測する品質管理に基づくプロセス改善を継続させていきたい．

第5章 結論

本研究は，1996年に筆者が品質保証部門の立ち上げを担当して以来2003年まで，品質保証と品質管理を直接担当してきた8年間の研究を基礎にして，高品質なソフトウェアを開発するために有効なプロジェクトマネジメント技術を，実際に改善・向上していく方法を体系的にまとめたものである．すなわち，プロジェクトマネジメント技術を導入し，マネジメントプロセスとしてソフトウェア開発組織へ定着させ，プロセスの定量的評価によりマネジメントの弱点を改善していくPDCAサイクルを確立する方法について議論した．

本研究の成果を以下にまとめる．

第2章では，実践的なプロジェクトマネジメント技術について，実際のプロジェクトへ導入しプロジェクトマネジメントを改善する方法を示した上で，プロジェクトのQCD目標の達成に対する有効性を分析し，より有効な活用方法を議論した．成果として，プロジェクトのリスクを早期に見極め軽減するリスク管理の有効性を示した．また，進捗管理では，EVM分析回数の粒度を密に行うマネジメントにより，プロジェクトのリスクを軽減し，QCD目標を達成することが可能になることを示した．さらに，ソフトウェア品質特性を用いたテスト項目によりテストを行い，ソフトウェア信頼性評価技術を用いて品質到達レベルを評価することにより出荷後のフォールトの発生を無くすることができることを示した．

第3章では，上記のプロジェクトマネジメント技術を定着させるための仕組みとして，ソフトウェアプロセスの改善技術について議論した．実際のソフトウェア開発組織の弱点を分析して，ソフトウェア開発プロセスとマネジメントプロセスを改善した事例を示した．また，ソフトウェアプロセスモデルCMMについて，考え方を説明し，CMMを用いたプロセス改善方法と改善効果について示した．さらに，標準ソフトウェアプロセスの実例を示した．

第4章では，プロセス改善を推進するために，プロジェクトマネジメントがソフトウェア製品品質や開発コストに及ぼす影響を評価する手法として，実際のプロジェクトにおける測定データを重回帰分析による予測モデル式を用いて定量的に分析することにより，プロジェクトマネジメントの改善すべき弱点を明確にする評価手法を提案した．

この定量的評価法は，導入し定着させたマネジメント技術の有効性確認と改善す

るべき課題を抽出することに有効であることを示した．定量的評価の結果，リスク管理によるプロジェクトマネージャへの支援活動が，プロジェクトを成功に導くために有効であることを確認した．また，プロセス品質を保証するレビュー活動の計画と進捗管理が，製品品質保証活動よりもソフトウェア製品品質の向上に有効であることを確認した．すなわち，本論文で提案したソフトウェアプロジェクトの定量的評価法は，プロジェクトマネジメントの弱点を改善・向上していくPDCAサイクルを，推進していくために有効な手法であると考える．

次に，今後の課題と展望を述べる．

本研究の成果は，CMMによって示されているレベル４の成熟度と同レベルであると認識している．今後は，４章で提案した定量的評価法の適用範囲を拡大し発展させることにより，持続的にマネジメント技術を改善するプロセスを構築していく必要がある．よって，定量的なプロセスのデータにより，持続的にマネジメント技術を改善するプロセスの研究が今後の課題である．

上記の課題を克服するために，より精度の高いリスクの評価方法，規模見積り精度の向上のためのファンクションポイント法[7],[12]などに取り組み，弱点プロセスの見極めに有効なメトリクス測定などのプロジェクトの定量的評価方法について考察しなければならない．定量的評価法により，プロセスとマネジメント技術の改善を推進し，その改善がプロジェクトに与える効果を見極める必要がある．

本研究で提案したソフトウェアプロジェクトの定量的評価法から，プロジェクトのマネジメント技術が持続的に改善されるようになってこそ，高品質ソフトウェアを開発するために必要なプロジェクトマネジメント技術を，常に改善・向上していく仕組みが完成すると考える．

参考文献

[1] T.Demarco and T.Lister: *Walting with Bears – Managing Risk on Software Projects*, 日経BP社 (2003)

[2] PMBOK2000翻訳プロジェクトチーム（訳）：プロジェクトマネジメント知識体系ガイド(PMBOKガイド)-2000年版，PMI東京支部 (2000)

[3] 能澤徹：国際標準プロジェクトマネジメントPMBOKとEVMS，日科技連出版社 (1999)

[4] 冨永章：解説アーンド・バリュー・マネジメント， プロジェクトマネジメント学会 (2003)

[5] 日本工業標準調査会：JISX0129 (ISO/IEC9126)，日本規格協会 (1994)

[6] 山田茂：ソフトウェア信頼性モデル―基礎と応用―，日科技連出版社 (1994)

[7] 山田茂，高橋宗雄：ソフトウェアマネジメントモデル入門―ソフトウェア品質の可視化と評価法，共立出版 (1993)

[8] Software Engineering Institute, Carnegie Mellon University (ed.): *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley Longman (1994)（アンダーセンコンサルティング (監訳) ：成功するソフトウェア開発―CMMによるガイドライン，オーム社 (1998)）

[9] J. Raynus (富野壽監訳)：CMMによるプロセス改善入門，共立出版 (2001)

[10]SLCP-JCF98委員会（編）：共通フレーム９８ SLCP-JCF98（1998年版），通産資料調査会 (1998)

[11]真野俊樹，誉田直美：見積りの方法，日科技連出版社 (1993)

[12]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)

[13]W. S. Humphrey ：パーソナルソフトウェアプロセス技法　能力向上の決め手，ソフトウェア品質経営研究会訳，共立出版 (1999)

[14]K. Esaki, S. Yamada, and M. Takahashi: "A Model for Reliability Prediction Based on Software Development Process Characteristics and Its Evaluation", *Proc. Second World Congress for Software Quality (2WCSQ)*, pp.813-818 (2000)

[15]山田茂，松田僚太郎："ソフトウェア設計レビューにおける信頼性に影響を及ぼす人的要因の品質工学的評価"，日本経営工学会論文誌，Vol.54,No.1, pp.71-79 (2003)

謝辞

本研究を遂行するにあたり，多大のご指導・ご鞭撻を頂きました鳥取大学工学部社会開発システム工学科山田茂教授，河合一教授，ならびに知能情報工学科池原悟教授に深く感謝いたします．特に指導教官の山田茂教授には鳥取大学大学院工学研究科博士課程に在学中，懇切なご指導と格別のご配慮，暖かい励ましを賜り，心から感謝しております．

本研究を進めるにあたり，様々な面でご支援を頂きました鳥取大学工学部社会開発システム工学科得能貢一助教授をはじめ，山田研究室の方々に深く感謝申し上げます．

本研究は，㈱日新システムズで品質保証部門を立ち上げ，その後の品質管理部での業務が基礎となっております．並々ならぬご配慮ならびにご支援を頂いた中村前社長，ならびに大西取締役に深く感謝申し上げます．共にソフトウェア品質の課題に取り組み，ご支援いただいた藤野孝司氏，小林克彦氏，ならびに江崎美保氏はじめ，関係者各位に改めて感謝申し上げます．

日本科学技術連盟内の高品質ソフトウェア技術交流会 (QuaSTom)では，講演会，分科会，合宿にて多くのことを学ばせて頂きました．ご指導いただいた多くの諸先輩や幹事の皆様方にこの場を借りて感謝いたします．

最後に，鳥取大学大学院工学研究科博士課程で学ぶことを喜び多くの支援をしてくれた父秀夫，母千代子に感謝します．日頃から心の支えとなり励ましてくれた妻千枝美，ならびに娘望，息子広樹，悠太に感謝します．

研究業績一覧

主論文

1. 山本美保, 福島利彦: “プロジェクト成功のためのリスク管理活動事例”, ソフトウェアシンポジウム 2000 論文集, pp.194-196 (Jun. 2000).

2. M.Yamamoto, T.Fukushima, and S.Yamada: “Software Process Improvement for Quality Product Development –Process and Product Quality Evaluation based on a Software Reliability Assessment Tool”, *Proc. the Second World Congress for Software Quality (2WCSQ)*, pp.237-242 (Sep. 2000).

3. 江崎美保, 福島利彦, 山田茂: “成功するソフトウェア開発のためのソフトウェアプロセス改善活動”, プロジェクトマネジメント学会誌, Vol.3, No.2, pp.15-20 (Apr. 2001).

4. 福島利彦, 江崎美保, 小林克彦, 山田茂: “リスク管理活動によるソフトウェアプロセス改善”, 第20回ソフトウェア生産における品質管理シンポジウム発表報文集, pp.265-272 (Nov. 2001).

5. 福島利彦, 山田茂: “ソフトウェアプロセス改善への取り組み—EV 分析を用いた進捗管理事例—”, 第21回ソフトウェア生産における品質管理シンポジウム発表報文集, pp.71-78 (Dec. 2002).

6. 福島利彦, 山田茂: “ソフトウェア品質とプロセス品質向上のための持続可能なプロジェクトマネジメント”, SEPG Japan 2003 予稿集, pp.135-140 (Sep. 2003).

7. 福島利彦, 山田茂: “プロジェクトマネジメント技術の持続的向上による QCD 効果”, 第 22 回ソフトウェア生産における品質管理シンポジウム発表報文集, pp.201-208 (Dec. 2003).

8. 福島利彦, 山田茂: “Continuous Improvement Activities in Software Process”, 情報処理学会ソフトウェアジャパン 2004 論文集, pp.9-14 (Oct. 2004).

9. T.Fukushima and S.Yamada: “The EVM and Its Effect to Software Project Development”, *Proc. Second International Conference on Project Management*

(*ProMAC2004*), pp.665-670 (Oct. 2004).

10. 福島利彦, 山田茂: "プロセスデータを用いた早期ソフトウェア製品品質の予測", 第23回ソフトウェア生産における品質管理シンポジウム発表報文集, pp.187-194 (Dec. 2004).

11. T.Fukushima, A. Fukuta and S.Yamada: "Early-Stage Product Quality Prediction by Using Software Process Data", *Proc. the Eleventh ISSAT International Conference on Reliability and Quality in Design*, pp.261-265 (Aug. 2005).

12. 福島利彦, 山田茂: "高品質ソフトウェアのためのプロジェクトマネジメント", 日本信頼性学会誌, , Vol.27, No.7, pp.483-494 (Oct. 2005).

参考論文（著書）

1. 福島利彦, 山田茂: "第3部第3章1 CMMでソフト開発プロセスを改善", プロジェクトマネジメント大全, プロジェクトマネジメント学会・日経コンピュータ（編）, 日経BP社, pp.188-193 (Oct. 2002).

2. 福島利彦, 江崎美保, 山田茂: "第 10 章日新システムズにおけるソフトウェアプロセス改善と品質保証の実際", ソフトウェアプロセス改善と品質保証の実際, 瀧本（監）, 日本テクノセンター, pp.311-340 (Oct. 2004).

参考論文（研究報告・発表）

1. 山本美保, 福島利彦, 並河安則, 山田茂: "ツール導入によるソフトウェア品質の可視化～ソフトウェア信頼度成長モデルの適用と評価～", 日新電機技報, Vol.44, No.2, pp.85-90 (Jul. 1999).

2. 福島利彦: "CMM によるプロセス改善の方法と課題",高品質ソフトウェア技術交流会(QuaSTom), 講演資料 (Jun. 2001).

3. 福島利彦: "Earned Value によるソフトウェア開発プロジェクトの管理事例", 高品質ソフトウェア技術交流会(QuaSTom), 講演資料 (Jul. 2002).

4. 福島利彦: “日新システムズにおける品質向上への取組み”, 技術講座「定量的に管理するためのソフトウェアの信頼性評価技術とその実際」(日本テクノセンター), 講演資料 (Nov. 2002).

5. 福島利彦: “ソフトウェアプロセス改善活動実践事例”, ソフトウェア品質技術実践講座「ソフトウェアプロセス改善実践コース」(日本科学技術連盟), 講演資料, pp.160-183 (Jan. 2003).

6. 福島利彦: “日新システムズにおける SPI 活動実践事例”, ソフトウェア品質技術実践講座「ソフトウェアプロセス改善実践コース」(日本科学技術連盟), 講演資料, pp.149-169 (Jun. 2003).

7. 福島利彦: “SPI 活動実践事例”, ソフトウェア品質技術実践講座「ソフトウェアプロセス改善実践コース」(日本科学技術連盟), 講演資料, pp.141-159 (Dec. 2003).

8. 福島利彦: “ソフトウェアプロセスの持続的改善と QCD 効果”, ソフトウェアプロセス改善シンポジウム発表報文集(日本オペレーションズ・リサーチ学会中国・四国支部), pp.14-21 (Feb. 2004).

9. 福島利彦: “ソフト開発リスクへの挑戦-品質管理部門によるリスク管理活動事例-“, SPC ミニセミナー (日本科学技術連盟), 講演資料 (Feb. 2004).

10. 福島利彦: “日新システムズにおけるプロセス品質向上への取組み”, 技術講座「高レベルな品質を追求するためのソフトウェアの信頼性評価技術とその実際」(日本テクノセンター), 講演資料 (Sep. 2004).

11. 福島利彦: “高品質ソフトウェア開発のためのプロジェクトマネジメント技術—アーンド・バリュー・マネジメント—”, 第 2 回プロジェクトマネジメントと最適化研究部会(日本オペレーションズ・リサーチ学会中国・四国支部), 講演資料 (Oct. 2005).

END