企業におけるソフトウェア品質向上に向けた
プロセス改善に関する研究

小笠原 秀人

電気通信大学 大学院 情報システム学研究科

博士（工学）の学位申請論文

2011年12月

企業におけるソフトウェア品質向上に向けた
プロセス改善に関する研究

博士論文審査委員会

主査　大須賀 昭彦　教授
委員　渡辺 俊典　　教授
委員　鈴木 和幸　　教授
委員　田中 健次　　教授
委員　田原 康之　　准教授

A Study for Promoting Software Process Improvement in a Large-Scale Organization

Hideto Ogasawara

Abstract

While the technical advancements continue in Software Engineering, mistakes and failures such as schedule delays, accidents due to software quality and wrongly estimated costs while are caused by how software development projects are managed remain as critical issues. The concepts of "software process" appeared on the scene in order to counter these problems. Since then, numerous debates have taken place and various technical solutions have been proposed but, root causes for these failures in software development projects still remain unimproved. In particular, there is a problem of being unable to implement improvement because the software processes that have been defined for fixing the root cause, are not being used in the development projects.

This paper shows the actual results and effectiveness of mechanism built based on process management area of CMMI for promoting process improvement activities in a company which consists of a large number of development departments. The mechanism promotes the process improvement activities, and contributes to the improvement in process maturity of each development department greatly. Given this situation, it is clear that the quality of the mechanism has a great influence on overall improvement activities, and that it is important to find the most effective method of constructing the mechanism. For nearly ten years, we have been constructing the mechanism and promoting process improvement activities using it. As a result, maturity levels was improved in a large number of development departments, where the applicability of the mechanism has been confirmed.

The value of this paper is having proposed and built the mechanism for the practical

software process improvement activity promotion in a large-scale organization. And, it is in the practicality of the built mechanism having been shown. By referring to this paper, the promotion method of software process improvement activities and important matter in a company should be understood, and it must be useful for more practical planning and enforcement.

企業におけるソフトウェア品質向上に向けた プロセス改善に関する研究

小笠原 秀人

概要

ソフトウェアがさまざまな領域で活用されるのにともない，製品やシステムにおけるソフトウェアが果たす役割は増している．1990 年代以降，製品やシステムの高機能化・多機能化にともない，ソフトウェアの大規模化・複雑化が急激に進んでいる．さらに，ユーザニーズが多様化してきたことによって，製品サイクルの短縮，製品ファミリーの拡大といった傾向も顕著になってきた．

このような状況に対応しきれず，ソフトウェア開発の問題が発生し続けている．多くの企業では，ソフトウェア開発における QCD（Quality : 品質，Cost : コスト，Delivery : 納期）を安定させるために，ソフトウェアプロセス改善活動を推進している．

プロセス改善活動を推進するために，多くの企業では，CMMI（能力成熟度モデル統合 : Capability Maturity Model Integration）や ISO9001 などのモデルを活用している．しかしながら，それぞれのモデルでは，具体的にどのように実施すればよいかということは記述されていない．そのため，プロセス改善活動は推進しているものの，活動自体が形骸化してしまう，過重なプロセスを作ってしまい品質向上や生産性向上に寄与できていない活動になってしまう，という結果も報告されている．つまり，これらのモデルだけでは，プロセス改善活動を推進するには十分なものが揃っているとはいえない，ということを示している． さらに，プロセス改善活動の中で導入・推進が図られるソフトウェア品質向上のための管理手法や管理ツールに関しても，組織内への導入と定着のための方法が明確でないため，先行部門ではうまく導入され定着するものの，組織横断的な展開には失敗するということも大きな問題として認識されている．

これらの問題を解決するため，本研究では，企業におけるソフトウェアプロセス改善活動を全社的に実践することを目的として，ソフトウェアプロセス改善活動を推進・定着さ

せるための「SPI[1]フレームワーク」を提案し，その提案に基づいて構築した東芝版 SPI フレームワークを活用して大規模な組織[2]に対して約 10 年間にわたり実践した．構築したフレームワークの中には，ソフトウェア品質向上のために活用される管理手法や管理ツールを，組織横断的に展開するための方法の提案も含まれている．

大規模組織における約 10 年間にわたる実践の結果，構築した東芝版 SPI フレームワークが全社的に認知され，機能し続けているという状態を達成した．また，全社的な推進活動の成果として，組織成熟度が高くなり，ソフトウェア開発における QCD の向上にも貢献できていることを示した．

本論文の価値は，組織横断的に SPI 活動を推進・展開させるための「SPI フレームワーク」を提案し，その提案に基づき，東芝内の状況に合わせて東芝版 SPI フレームワークを構築したことと，その構築したフレームワークの実用性，汎用性を示したことにある．本論文を参照することによって，企業におけるソフトウェアプロセス改善活動の推進方法や留意点がわかり，より具体的な計画立案と実施に役立つはずである．

[1]SPI:Software Process Improvement．ソフトウェアプロセス改善は SPI と呼ばれることが多い．本論文でも，ソフトウェアプロセス改善活動のことを，SPI 活動と表現する．

[2]本論文では，おおよそ 50 以上の開発部門を持つ組織を大規模組織と考えている．

目 次

図 目 次

表 目 次

第1章　はじめに

本章では，本研究の背景と目的を述べたうえで，ソフトウェアプロセス改善[1]における本研究の位置付けを整理する．

1.1　本研究の背景

1960 年代半ば，メインフレームコンピュータが普及するとともに起こったソフトウェア危機 [1][2] を解決する手段として，1968 年，NATO（北大西洋条約機構）の技術委員会の主催によりドイツで開催されたソフトウェア工学会議の準備会議において，ミュンヘン工科大学の Bauer により「ソフトウェア工学」という概念が提案されたと言われている．その後，要求分析，ソフトウェア仕様，設計法，検証・保守といった，ソフトウェアライフサイクルの各開発段階の明確化の研究が進められ，データの抽象化，構造化，モジュール化などの方法論や各種技法の研究，さらにソフトウェア生産の自動化を目指す CASE（Computer Assisted(Aided) Software Engineering）などの開発環境が提案され現在に至っている [3].

ソフトウェア工学の技術的な進展の一方で，ソフトウェアの開発規模の増大に伴い，見積りの不正確さ，納期の遅れ，ソフトウェア品質に基づく事故など，ソフトウェア開発プロジェクトの管理の側面での失敗が問題となってきた [34][4]．大規模ソフトウェア開発プロジェクトの管理という，ソフトウェア開発が抱えるようになった新たな課題に対し,「ソフトウェアプロセス」という概念が登場し，広く議論されるようになったのは 1984 年にイギリスの Egham で行われたソフトウェアプロセス国際ワークショップ（ISPW:International Software Process Workshop）からのことであると言われている [5][6]．そして 1987 年，ソフトウェア工学国際会議（ICSE:International Conference on Software Engineering）の第 9 回大会において米国の Osterweil が,「ソフトウェアプロセスもソフトウェアである（Software Process Are Software Too）」という論文 [7, Osterweil] を発表して大きな反響を呼び，ソフトウェアプロセスの概念が急速に広まった．

今日，製造，流通，販売，そしてサービスといった産業全般にわたって，ソフトウェアが果たす役割は益々広がっている．ソフトウェアを開発する企業にとっては，高いソフトウェア品質が市場競争における重要な要素となり，また，日本の製造業を代表する組み込

[1]SPI : Software Process Improvement の略．ソフトウェアプロセス改善は SPI と呼ばれることが多い．本論文でも，ソフトウェアプロセス改善活動のことを，SPI 活動と表現する．

み系ソフトウェアの開発を行っている企業にとっても，ソフトウェア開発の技術力が企業の競争力を上げるための重要な課題となっている．このような産業構造のソフトウェア比重の増大に伴い，ソフトウェア開発の技術力が産業全体に大きく影響を及ぼすようになりつつあり，この傾向は今後益々増してくると予想される．その一方で，ソフトウェアの大規模化に伴ったソフトウェア開発のコストやリスクも増大する傾向にあり，ソフトウェアプロジェクトの失敗は，企業のビジネスに大きく影響する重要な課題となっている．1960 年代半ばにメインフレームコンピュータの普及とともに起こったソフトウェア危機は，ソフトウェア規模の増大，企業競争の激化に伴う品質や生産性競争として形を変えて現在に至っている．

このようなソフトウェアの品質や生産性の課題を解決するために，米国では，米国国防総省（DoD : Department of Defense）のソフトウェア調達モデルとして開発された CMM/CMMI[2]が，ソフトウェア開発プロセス改善のための参照モデルとして利用され，主にソフトウェアプロセスの管理面における適用の研究が急速に進んだ．このモデルは変化し続けており，このモデルの初期のバージョン CMM から現在では，CMMI へと進化している．一方，急激な規模の拡大や企業間の競争力として重要となったソフトウェア開発に対して，品質・コスト・納期（QCD : Quality，Cost，Delivery）に関する課題を抱えていた日本の企業においても，1998 年頃より，CMM によるプロジェクトの管理技術に着目したプロセス改善の議論がはじまった [12][13]．2002 年 4 月には，プロセス改善に関する議論の場として日本 SPI コンソーシアム（JASPIC）が設立された．現在，より高い生産性の実現や市場の要求の変化に対して，より柔軟な対応能力が求められており，プロセス改善はますます重要な課題となっている [8]．

しかしながら，人的リソース不足，プロセス改善のための技術力不足や能力成熟度モデルの活用方法に関する理解不足などにより，組織的に継続してソフトウェア開発における QCD 向上のプロセス改善活動を効果的・効率的に実践することが困難なケースも少なくない [15][16]．特に短期的な成果を求めがちな昨今の経営環境にあって，中長期的な経営戦略の中で，プロセス改善を間断なく進めていくことは非常に難しい．例えば納期遅延を防止するためのプロジェクト管理ツールを導入したとしても，自分たちの製品の特徴に合致した見積り基準や進捗管理のための閾値を，過去のプロジェクトの実績値の収集と分析から構築していかなければ，単にお絵書きツールに終わってしまい，真に有効なプロジェクト管理ツールとはなりえない．またそのツールの活用が場当たり的にならないためには，組織としてプロジェクト管理全般の教育を必修化するなどの施策も重要である．このように，ツールの活用ひとつとっても，その周辺には組織として取り組むべき内容が多く存在していることが分かる．それゆえ，組織として中長期的な技術施策の中で，ステップを踏

[2]CMM:Capability Maturity Model, CMMI:Capability Maturity Model Integration．ソフトウェア開発プロセスの参照モデルのデファクトスタンダード．5 段階で組織の成熟度を評価する．

みながらプロセス改善を進めていくことが必要である.

1.2 本研究の目的

ソフトウェアプロセスの持続的な改善を推進する方法論として，能力成熟度モデル CMMI[36] が多くの組織で活用されている．この CMMI を利用し，SPI 活動をより効果的・効率的に実践するために，文献 [17, 福山] では，推進体制や普及手段など 7 種類の広義の支援ツールが提案され，その支援ツールの適用による効果も示されている．また，組織的に推進するための方法論として，文献 [18, IDEAL] で IDEAL[3]モデルが提案されている．組織のあるべき姿として CMMI を利用した SPI 活動を推進する際には，多くの場合，この活動を推進する専門の部門が，文献 [17, Fukuyama] や文献 [19, 福山] で示したような支援ツールを準備し，複数の開発部門へ同時並行に展開することを計画する．同時並行に展開するために，一般的には，ひとつの部門で先行して試行し，そこで得られた知見をまとめ，他の部門へ展開するというアプローチを取ることが多い.

しかしながら，試行対象の部門で成功した結果を他の部門へ展開する際には，試行時のような手厚い支援を多くの部門に同時並行で提供できないという難しさがある．企業の中で品質管理活動やプロセス改善活動を展開することの難しさは，文献 [20, Gargi Keeni] で述べられている．また，筆者らは，1990 年代，ソフトウェア品質管理のための手法やツールを全社的[4]に推進する際に [21][22][23][24]，組織横断的に技術を移転することの難しさを経験している.

大規模組織において SPI 活動を展開・推進するものの，期待された効果が得られない原因は，以下のように考えられる.

- **原因**

  ソフトウェア開発のベストプラクティスである CMMI が広く普及しているが，それを利用して組織横断的に SPI 活動を推進するための方法論が確立されていない.

このような状況の中，企業においては，組織横断的に SPI 活動を確実に，そして効果的・効率的に実現する方法が求められている.

そこで，本研究では，上述した SPI 活動推進を阻害する原因の解決を目的とし，以下の項目に関する研究を行った.

- **目的**

  企業において組織横断的に SPI 活動を推進・展開するための方法論の確立

[3]プロセス改善を推進するためのモデルであり，Initiating(開始)，Diagnosing(診断)，Establishing(確立)，Acting(実践)，Learning(学習) の 5 つのフェーズから構成されている.

[4]本論文では，全社という用語を，大規模組織全体を示す意味で使用する.

研究は，筆者が企業内において SPI 活動を全社的に推進・展開するミッションを持っていたという利点を生かした．具体的には，SPI 活動を推進・定着させるための仕組みとして「SPI フレームワーク」を提案し，この SPI フレームワークに基づいて東芝版 SPI フレームワークを構築した．さらに，これを活用して東芝グループ内で SPI 活動を約 10 年間の長期にわたり実践した．その実践結果をまとめ，提案した SPI フレームワークの汎用性，SPI フレームワークに基づいて構築した東芝版 SPI フレームワークの実用性と有効性に関して評価した．

1.3　本研究の位置付け

ソフトウェアプロセス改善活動カンファレンス (SEPG North America，EuroSPI，SPI Japan など) に代表されるソフトウェアプロセスに関する会議では，主に成熟度毎の改善活動の推進方法，改善活動を推進するために設置されるグループの役割や機能，モチベーションの維持，管理手法や管理ツールの導入・展開方法，プロセス改善の効果などに関しての研究や議論，紹介が行われている．しかしながら，従来の対象範囲は，期間については最大で数年というスパンが多く，規模に関しては，最大でも事業部レベルというものが多かった．全社的な SPI 活動の普及・展開を考えると，対象とする組織数が多く，長期間にわたる活動に対するアプローチが必要となる．

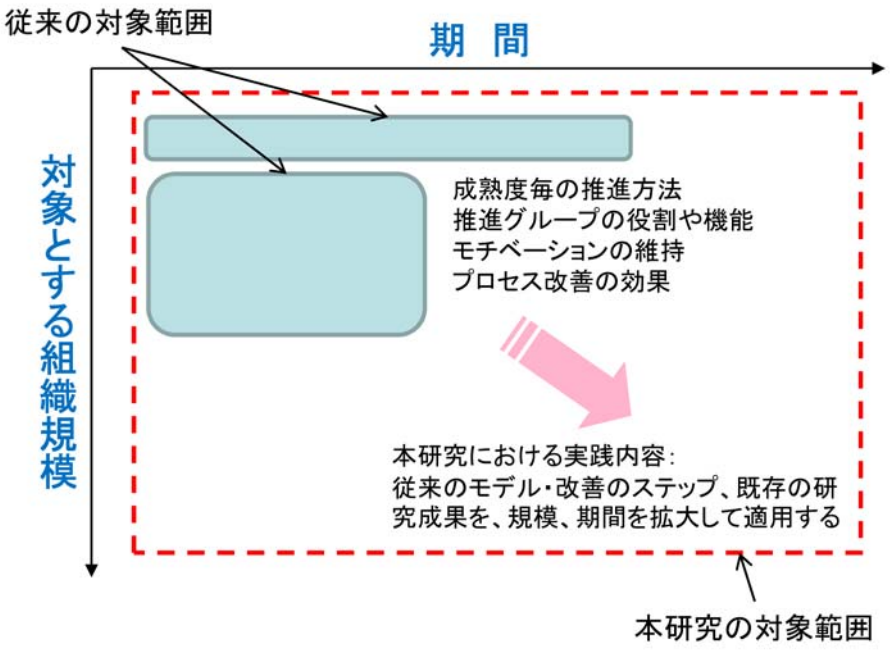


図 1.1: 本研究の位置付け

本研究は，大規模組織において，既存のモデル，改善のステップ，研究成果などを利用

したSPI活動の推進方法に関する研究である．既存の研究と本研究の関係を図1.1に示す．

図1.1に示すとおり，本研究は既存の研究とは違う領域に対するアプローチであり，既存のモデルや改善のステップ，研究成果と大規模組織におけるSPI活動の推進・展開という活動の間を支えるための研究であるといえる．

1.4 論文の構成

本論文の構成は以下のとおりである．第1章では，ソフトウェアプロセス改善を取り巻く課題を整理し，本研究の目的と位置付けを明確にする．第2章で，ソフトウエア開発における課題と既存の研究成果を示す．第3章では，SPI活動を推進・定着させるための枠組みである「SPIフレームワーク」を提案し，こｐのフレームワークに従って構築した東芝版SPIフレームワークの構成と内容を説明する．続く第4章で，第3章で示した東芝版SPIフレームワークを活用してSPI活動を約10年間の長期にわたり実践した内容とその評価結果を示す．さらに，第5章では，SPIフレームワークを構成する要素間の関係性，提案したSPIフレームワークの汎用性，東芝版SPIフレームワークを構築する際に得た知見，計画どおりに実践できなかった項目の原因について考察する．最後の第6章で本研究を総括し，課題に対して設定した目的の実現度合いや実践した内容の妥当性をまとめる．さらに，今後の進むべき方向性を述べる．

また，付録として，東芝版SPIフレームワークにおいて改善ソリューションとして提供してきた，ソフトウェア品質管理技術とプログラム静的解析技術に関する研究・開発の成果を紹介する．

第2章　ソフトウェア開発における課題と既存の成果

本章ではソフトウェア開発における現状の課題を確認後，この課題を解決するための基盤として研究・開発，提案されている既存の成果を整理する．

2.1　ソフトウェア開発の現状

高機能化，複雑化するソフトウェアを開発するうえで，全開発工程をとおした品質作り込みおよび品質管理の高度化が求められている．高機能化に伴い，ソフトウェアの規模やテストすべき機能が増える一方で，ソフトウェア開発期間は更なる短縮化が迫られる状況にある．

ソフトウェア開発における課題を表すデータとして,「組込みソフトウェア産業実態調査報告書[1]」) から，ソフトウェア開発の課題と不具合に起因する品質問題の再発防止策の集計結果を，それぞれ図 2.1[26] と図 2.2[25] に示す．

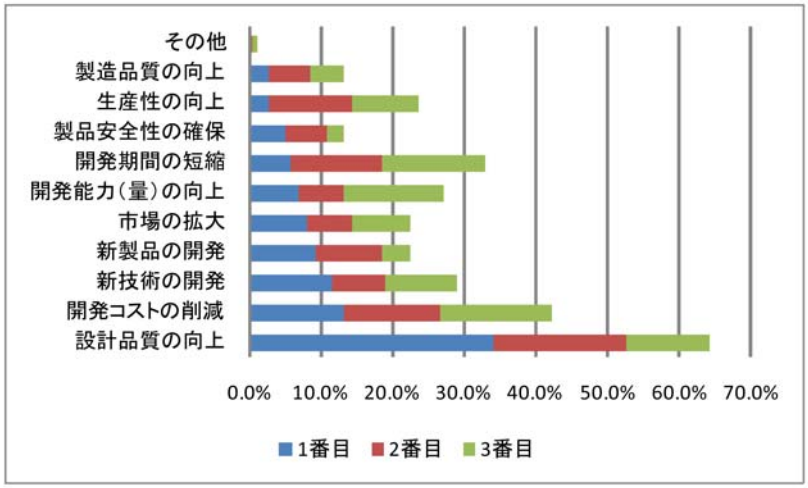


図 2.1: ソフトウェア開発の課題

図 2.1 から分かるとおり，ソフトウェア開発においては「設計品質の向上」,「開発コストの削減」,「開発期間の短縮」の QCD に関連した課題が上位に挙がっている．このよう

[1]経済産業省 商務情報政策局 情報処理振興課から毎年発行されている

な課題に対する解決策としては，図 2.2 に示すとおり,「ソフトウェア開発プロセスの見直し」,「技術者への品質管理教育の実施」,「品質管理基準・規定の策定・見直し」が上位の 3 つに挙げられている．

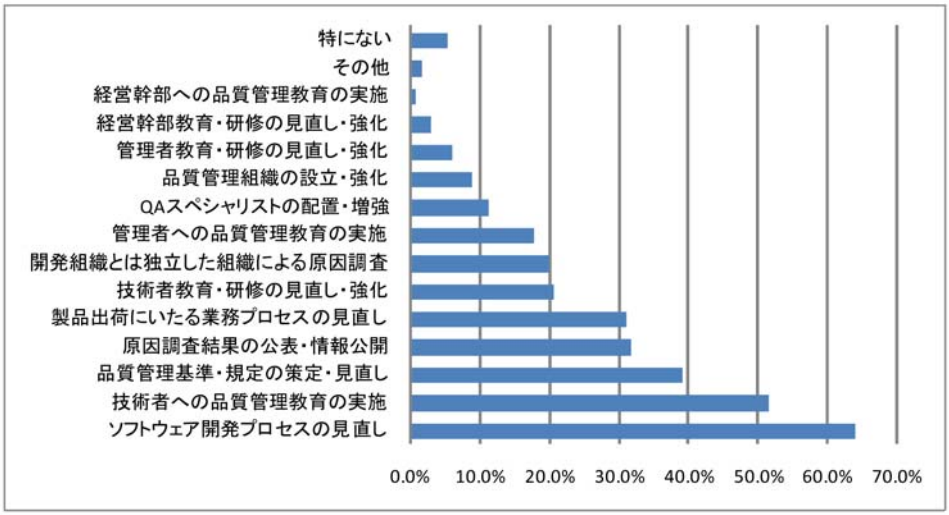


図 2.2: 不具合に起因する品質問題の再発防止策

ソフトウェア開発は反復型の学習プロセスである．プロセスに従って知識を集積し，抽出し，体系だてたものを具現化したものが「ソフトウェア資産」となる．この資産を確立し維持するために，多くの企業が SPI 活動に取り組み，図 2.2 で示したような再発防止策を推進している．

2.2　プロセス改善の本質

プロダクトとは，あるプロセスから生成される成果物をいう．プロセスには入力があり，その入力を元にして出力が生成される．また，プロセスには，その入力から出力を生成する変換過程を特徴付けるパラメータがある．したがって，同一の入力が与えられたとしても，変換過程としてのプロセスの特徴が変化すれば，出力である成果物も変化する．この考え方を模式的に表したものを図 2.3 に示す．ソフトウェア開発に置き換えてみると，顧客からの要求に対して，いくつかのプロセスを経てソフトウェアを開発し納品する．納品物としてはソフトウェアではあるが，それがどれだけの価値を顧客に提供できるかどうかは，それを作り出すプロセスの品質に依存する（図中の①に対応）．また，あるプロセスに着目すると，入力から出力へ変換するための活動があり，その活動を行うための資源が必要となる．さらに，その活動の状況を知るために，計測が行われ，適切な管理が必要となる（図中の②に対応）[10].

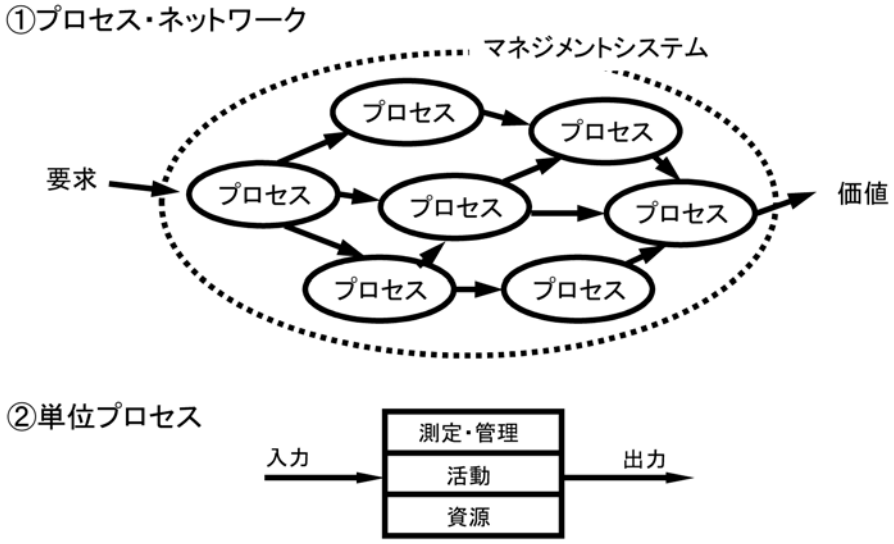


図 2.3: プロセスネットワークと単位プロセス

顧客に対する価値を最大にするためには，良い結果が得られるように方法を決めてそのとおりに実施することが効果的であり，かつ効率的である．もし，その方法に問題があれば，それを改善することが重要である．つまり，プロセスを管理するということである．

「プロセス管理の原則」を以下に示す.

- 基本的考え方
  - 工程で品質を作り込む
  - 結果に着目するだけでなく，結果を生むプロセスについて注目し，仕事のやり方を改め，仕事の品質を向上させる
- 望ましいプロセス条件に標準化する
  - 良い「結果」が得られるようなプロセス条件を明らかにする
  - 良い「結果」を得るためにプロセス中での確認事項を明らかにする
  - プロセス中で実施すべき事項を標準化する
- 標準どおりの仕事を行う
- 管理・改善
  - 目標と実績の差異の要因解析を行い，要因系を抑え込む
  - 良い結果が得られるようなプロセスの条件を見極め，現状の仕事のやり方を改善し，最も良い仕事のやり方に改めていく

上記の原則にしたがってプロセスを改善する.「プロセス改善の基本」を以下に示す.

- プロセスの維持＋ α
  - いついかなる時も，技術やマネジメントシステムは不完全
  - 不満足な状況が発生したら確実に解消
  - 小さな改善の積み重ねにより大きな進歩
- 全員参加の改善
  - 問題の真の姿は当事者が一番よく知っている
  - 現場の作業者・事務員レベルの問題意識・改善意識
- 第一線の従業員の自主性・主体性・積極性
  - 言われたことだけを忠実に行うという枠を越えてもよい
  - 自分の仕事と製品・サービスの質との間の関係の理解
  - 職場の中での自分の仕事の位置付けの理解
  - 業務において使用する方法論，手法，設備・機器，情報システムなどの原理，論理，構造などの理解

上述した「プロセス管理の原則」と「プロセス改善の基本」に基づき，プロセス改善活動を推進する．この活動では，適切なプロセスを確立し，適切にプロセスを実施し，プロセスの実施状況を適切に監視し，必要であれば適切な対応を取ることが重要となる．これを，PDCA（Plan-Do-Check-Act）のデミングサイクルと呼ぶ．デミングサイクルを確実に実施するためには，組織内において標準的なプロセスを定義し，それを適確に応用することが必要となる．デミングサイクルにおける各フェーズの定義を図 2.4 に示す．

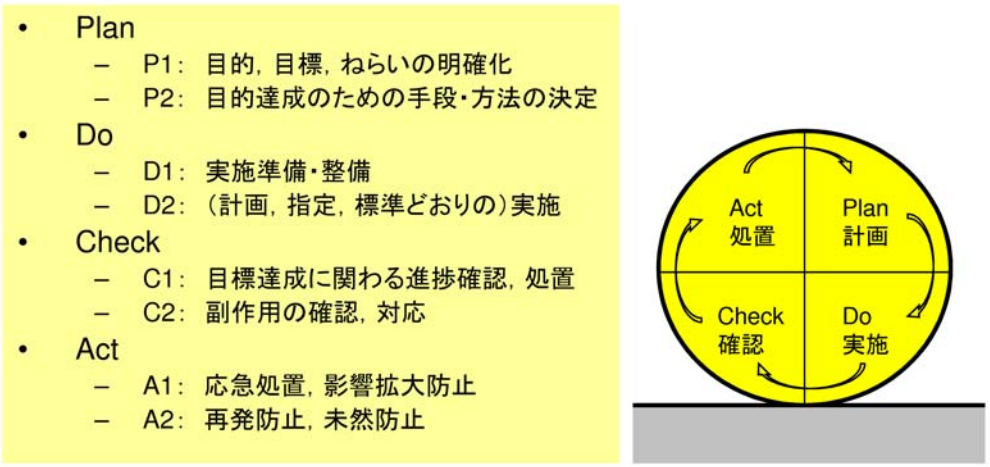


図 2.4: PDCA のデミングサイクル

2.3 「課題解決型」の改善と「改善モデル型」の改善

開発プロセスの改善方法は，以下に示すように,「手順のない改善活動」と「手順のある改善活動」が考えられる [11].

- 手順のない改善活動
- 手順のある改善活動
  - 改善モデルがない：課題解決型（QC ストーリー型）
  - 改善モデルがある：改善モデル型（診断改善型）

「手順のない改善活動」推進の問題点として以下の項目が挙げられる.

1. 活動範囲を絞りづらい
2. 現状を無視した的外れな改善提案をしてしまう
3. 場当たり的であり，流行の改善活動ばかりしてしまう
4. 第三者の改善提案だと対象組織のコンセンサスが得づらい
5. ツール主体になりがちである
6. 組織が変わるたびに，その場で改善活動のやり方を考えなければならない

「手順のある改善活動」の方法としては,「課題解決型」の方法である QC ストーリー的な手順と「改善モデル型」の方法である CMM のようなモデルを利用した診断を繰り返しながら推進する手順の 2 通りがある．改善活動のタイプ別手順を図 2.5 に示す.

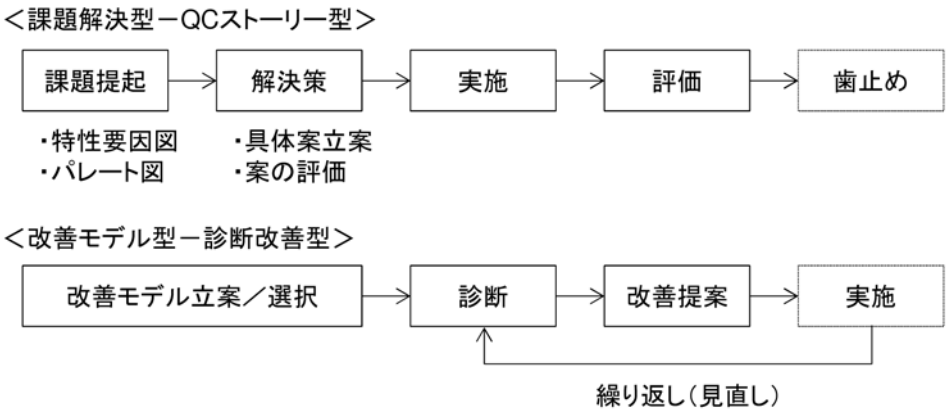


図 2.5: 改善活動のタイプ別手順

「手順のある改善活動」は，改善活動の手順があり，以下のようなメリットがあると考えられる.

1. 手順により活動が一貫している
2. 活動手順が分かるために，次の目標が定めやすい
3. そのため改善活動に対して組織のコンセンサスが得られやすい
4. また手順に従って，一過性ではなく継続して活動できる

改善活動のタイプ毎の特徴，長所，短所を表 2.1 に示す.

表 2.1: 改善活動のタイプ毎の比較

分類	特徴	長所	短所
課題解決型	・課題分析により解決が提案される ・ボトムアップ的活動が主	・確立した方法 ・短期的な解決 ・場に則した解決案	・一過性になりやすい ・組織の力に制限された解決案
改善モデル型	・診断結果に基づき改善提案が提出される ・トップダウン的活動が主	・モデルから一貫した改善内容 ・第三者が参加しやすい ・仮説検証サイクルが回る	・長期的な改善になってしまう ・モデル作りが困難 ・改善指標が重要になる

2.4 改善活動を推進する母体

プロセス改善というのは最終的には経営の効率を上げることにつながるものであるから，まず経営としての課題認識が大前提となる．もちろん，開発現場の個人が改題認識をして改善をスタートさせることもあるが，それだけでは改善成果には限界がある．プロセス改善にはしかるべき経営資源の投入が必要である．また経営としては経営資源の投資をするわけであるから，それによるリターン，つまり効果を何で計るかを定義しなければならない．この定義されるものには生産性・品質指標であったり計測精度であったりリワーク工数であったりするが，この定義された効果評価用の指標が経営から開発現場まで，縦通しで共通の目標として改善活動の拠り所となるものである．どのような改善活動のタイプをどのように自組織で適応していくのかを判断するのも組織としての取り組み方法に関する知識として重要なものである．さらにプロセス改善は組織文化の変革と組み合わされなければならないが，これをどのように有機的に結合させていくのかも組織として重要な知識である．

経営資源の投入によって組織化されるのが改善支援組織である．このメンバのことを一般的に SEPG（Software Engineering Process Group）と呼ぶ．プロセス改善の成否はこの SEPG がいかに効果的にプロセス改善を推進するかにかかっているといっても過言ではない．プロセスチャンピオンといわれるような強力なプロセス改善推進役の SEPG になるには高いヒューマンスキルと同時に以下のような知識が必要である．

まずプロセス改善の標準的なプロセスに関する知識と自部門への改善プロセスのカスタマイズ方法である．プロセス改善はやみくもに思いついた問題に対して取組んでいけばよいのではなく，その進め方にも標準的なプロセスが存在する．図 2.3 に示した，‘PDCA のデミングサイクルはその基本である．この改善サイクルのもっとも重要なところは現状プロセスの問題分析で，このためには現状プロセスの把握方法，その内容を記述するための表記方法，表記されたプロセスから改善点の抽出方法，その改善点の取組み優先付けといった知識が SEPG には要求される．また改善点抽出のベースとなり，さらに改善成果評価のために必須となる開発管理データの収集項目と収集方法についての知識が SEPG には要求される．つまりメトリクスに関する知識である．

SEPG にはこれらの知識だけではなく，次のような取組み姿勢が必要である．

- ソフトウェア課題と改善成果状況について常に上級管理職や経営層に適確に報告し，プロセス改善についての支持を常に維持していく．
- 開発現場と一体となって改善を推進し，単にプロセス変更のサポートをするだけではなく結果としての QCD の改善に責任を持つ．
- 開発現場の同意を得られないプロセス変更を強引に進めない．プロセス変更とプロダクトの改善の因果関係をデータで証明し，改善に関する知識を広くを吸収する．
- 自分達の改善成果を知識としてまとめると同時に，改善に関する知識を広く吸収する．
- SEPG はプロセス改善の知識で開発現場からの信頼を勝ち取らなければならない．また開発現場の対象ドメイン知識に対して敬意を払わなければならない．

このように SEPG には，非常に能力が高く改善に対して情熱を持って取組む人を割り当てなければならない．

SEPG は設計者に対して知識の提供をし，その知識を生かして設計者が改善成果を上げるようにするのを主たる役目としているが，場合によっては模範例として特定作業を分担し実行したり，設計者と共同作業を行い，直接的に改善成果を生み出すこともある．

SEPG が提供するのは，プロセスとプロダクトの QCD との因果関係の知識や，設計技法や開発環境に関する知識で，この知識を活用して設計者は効率的に設計を進めることができる．プロセス改善の成果は大半がここから得られるものである．

改善成果を上げるためには企業文化の改革，設計者の意識改革も大切な要素である．ソフトウェア開発は大変高度な知的作業であるので設計者の意欲，意識が生産性と品質に大きく影響してくる．つまりモラールの向上が改善成果に大きく影響するということであるが，このためにはまず経営からプロセス改善に対する明確な方針と指示が必要である．しかしこれだけであると設計者は改善の方法が分からないまま追い詰められる形になり，改善に対する意欲がそがれてしまうことになる．これを助けるのが SEPG の提供する改善に

関する知識で，この知識を活用すれば設計者自身も良い仕事ができ，さらに経営の要求を満足させることができ，設計者の改善に対するモラールを向上させることができる [9].

2.5 CMM の概要

前節で説明した「改善モデル型」の活動で広く活用されいている CMM は，米国国防総省 (DoD : Department of Defense) の要請でソフトウェア関連企業の能力評価のために，米国カーネギーメロン大学のソフトウェア工学研究所[2]によって 1993 年に提供されたモデルで，ソフトウェアの開発・保守組織が組織力の改善活動に用いるための，ソフトウェア開発プロセスの能力成熟度モデルである.

このモデルは組織の成熟度を 5 段階で定義しているところが特徴であり，ワッツ・S・ハンフリーの文献「Managing the Software Process」[31] をベースに，実際の成功例，失敗例を多数分析して作られた．最初に提示されてから，4 年間のレビュー期間を設け，現場のニーズと利用経験から改良されたものである．5 段階の成熟度の概要を図 2.6 に示す. CMM および CMMI の段階表現での各成熟度レベルの表現は多少異なり，かつ，各レベルを構成するプロセス領域などに違いがあるが，基本構造は同等である．CMMI が CMM をより洗練したものであるが，CMM の方が分かりやすいと考え，本章での説明では，基本的に CMM のモデルをもとに議論を進める.

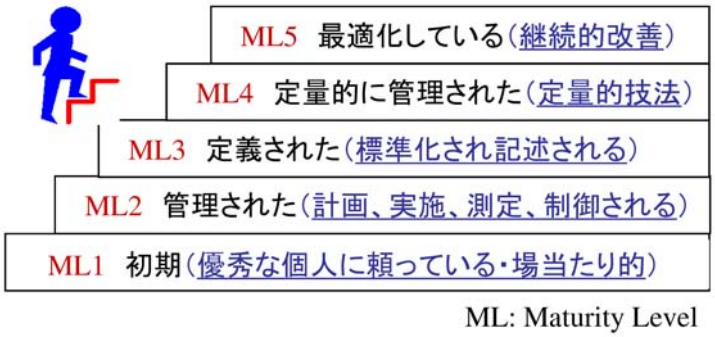


図 2.6: 各成熟度レベルの定義

レベル 2 では，個々のプロジェクトに対してプロセス管理の基本能力が求められており，この管理された状態によって，同等のプロジェクトにおいてプロセスを反復できる能力を持つようになる．レベル 3 では，一つひとつのプロジェクトでのプロセスの管理から，組織レベルのプロセスの管理能力に発展し，組織は，組織の標準となるプロセスを定義し，個々のプロジェクトで利用することが求められる．さらに，レベル 4 では，個々のプロジェ

[2]CMU/SEI:Carnegie Mellon University/Software Engineering Institute（カーネギーメロン大学 ソフトウェア工学研究所）

クトが実施したプロセスを定量的に評価することが求められている．レベル 2 から 4 の能力を獲得することによって，組織はプロセスを効果的・効率的に活用でき，さらに定量的な評価を行えるようになる．その結果，レベル 5 で定義されている“プロセスの最適化”が可能となる．

次に，このモデルの構造を図 2.7 に示す．

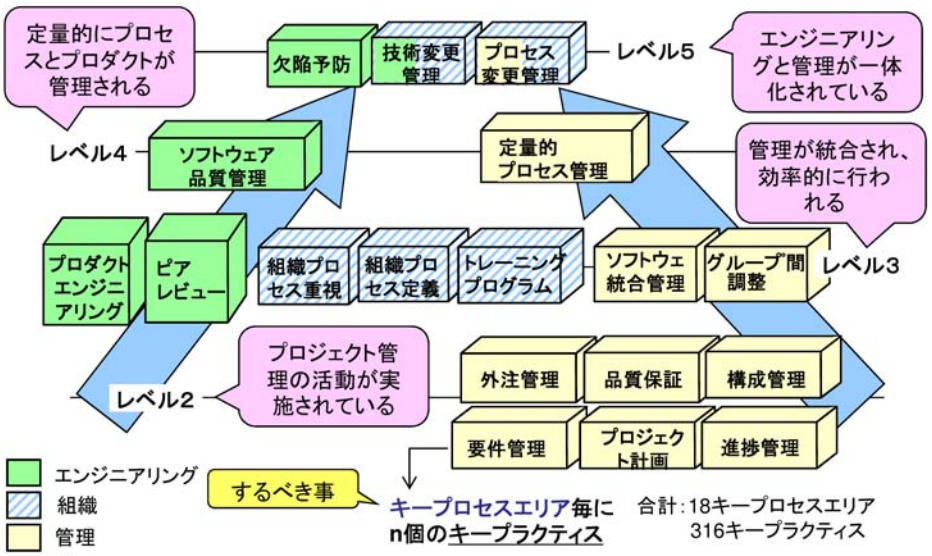


図 2.7: CMM の構造

各レベルに含まれるプロセス領域は，エンジニアリング，組織，管理の 3 つのタイプに分かれている．例えばレベル 2 には，要件管理，プロジェクト計画，進捗管理，外注管理，品質保証，構成管理の 6 つのプロセス領域があり，これらのプロセス領域は，管理のタイプに属するプロセスということが分かる．図中の吹き出しに，各レベルでの活動の概要を示す．個々のプロセスをキープロセスエリア (KPA:Key Process Area) と呼ぶ．キープロセスエリアには，複数のキープラクティス (KP:Key Practice) と呼ばれる“するべき事”が定義されている．ひとつのキープロセスエリアの内部構造を示した結果を図 2.8 に示す．キープロセスエリア毎に，2〜4 個のゴールと，複数のキープラクティスが定義されている．また，キープラクティスは，コミットメント，能力，計測，検証，活動という 5 つのタイプに分類されている．CMM 全体で見ると，18 個のキープロセスエリアがあり，その中で定義されているキープラクティスは，合計すると 316 個になる．

成熟度レベル 2 のキープロセスエリアである“要件管理”のゴールとキープラクティスを，表 2.2 に示す．表 2.2 に示したとおり，ひとつのキープロセスエリアには複数のキープラクティスが定義されている．さらに，ゴールを達成するためには，どのプラクティスが実践されなければいけないかという関係付けもされている．CMM を利用してソフトウェ

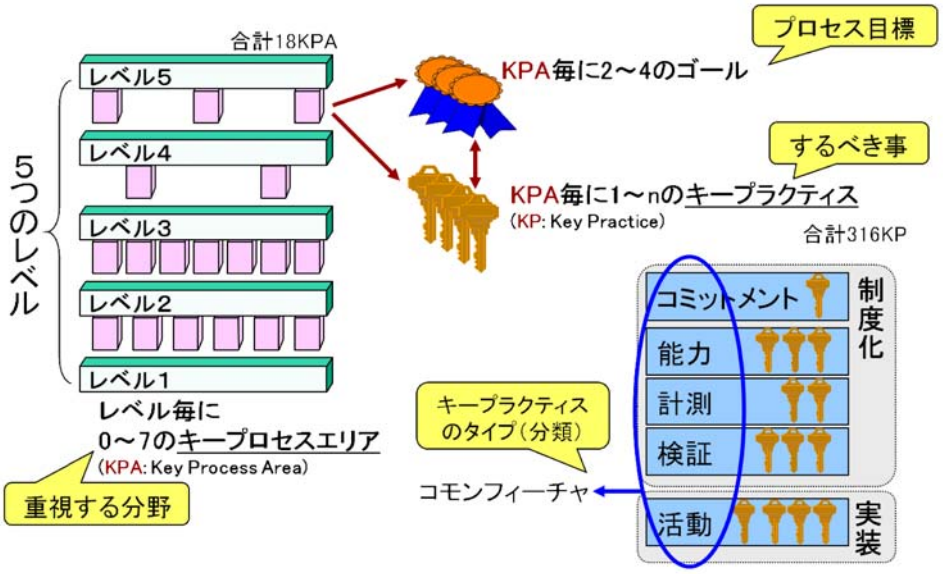


図 2.8: キープロセスエリアの内部構造

ア開発部門の成熟度レベルの評価をする場合には，組織の複数のプロジェクトに対してインタビューとドキュメントレビューが行われ，個々のプラクティスが実践されているかどうかが評価され，結果として，各キープロセスエリアのゴールが満たされているかどうか総合的に判断される．

表 2.2 では，ゴールとキープラクティスだけを抜粋したが，実際には，それぞれの項目についてより詳細な説明が付記されている．ページ数にすると，CMM，CMMI ともに，約 550 ページのボリュームがある．

2.6 プロセス成熟度の向上が意味すること

ハンフリーによって提案された能力成熟度モデル [31] はソフトウェア開発組織が信頼性の高い安定したプロセスをどうすれば作っていけるのかをモデル化したものである．その目的はソフトウェア開発組織の状況をプロセスの観点から分析し，その組織に合ったプロセス改善の目標を設定することにある [34]．このモデルはその後，能力成熟度モデル (CMM)Ver.1.1[34][35] として纏められ，ソフトウェアの品質・生産性向上のためのプロセス改善を進めるモデルとして米国のみならず世界の広い範囲で実質的なプロセス改善指針となりつつあり,「プロセスに着目した組織の能力および体質の改善が結果として組織全体の生産性ならびに品質の向上に寄与する」という考えを提唱している．

CMM は組織としての成熟度に焦点を当て，Q，C，D の不満足な結果を出す未成熟な組織と，満足な結果を出す成熟した組織を 5 段階に分類し，各段階を上がっていくための

表 2.2: 要件管理プロセスのゴールとキープラクティス

ゴール1	ソフトウェアのエンジニアリングと管理に使用するベースラインを確立するため、「ソフトウェアに割り当てられたシステム要件」が制御されている。
ゴール2	ソフトウェアの計画、成果物、および活動が、「ソフトウェアに割り当てられたシステム要件」と首尾一貫した状態に保たれている。

キープラクティス(KP)	キープラクティスの内容
コミットメント1	プロジェクトは、「ソフトウェアに割り当てられたシステム要件」の管理に関して、明文化された組織方針に従う。
能力1	各プロジェクトに対し、システム要件を分析し、それらをハードウェア、ソフトウェア、および他のシステムコンポーネントに割り当てる責任が確立される。
能力2	割り当てられた要件が文書化される。
能力3	割り当てられた要件を管理するために、適切な資源と資金が提供される。
能力4	ソフトウェアエンジニアリンググループや他のソフトウェア関連グループのメンバは、要件管理活動を実施するためのトレーニングを受ける。
活動1	ソフトウェアエンジニアリンググループは、割り当てられた要件がソフトウェアプロジェクトに組み込まれる前の段階でレビューする。
活動2	ソフトウェアエンジニアリンググループは、ソフトウェア計画、作業成果物、および活動の基盤として、割り当てられた要件を使用する。
活動3	割り当てられた要件への変更をレビューし、ソフトウェアプロジェクトに組み込む。
計測1	計測を行い、その結果を使用して、割り当てられた要件管理の活動状況を判断する。
検証1	割り当てられた要件の管理活動は、上級管理層によって定期的にレビューされる。
検証2	割り当てられた要件の管理活動は、プロジェクトマネージャによって定期的に、かつイベント発生を契機としてレビューされる。
検証3	ソフトウェア品質保証グループは、割り当てられた要件の管理活動と作業成果物をレビューかつ／または監査し、その結果を報告する。

注力点としてキープロセスエリアを定義している．

図 2.9 に成熟度の向上と Q，C，D プロセス実績の関係を示す．図の横軸は，Q，C，D の満足度を表しており，右に行くほど不満足な結果を表している．縦軸はある組織が複数のプロジェクトを実行した時のプロセス実績の分布を表している．未成熟な組織においては達成不可能な計画値で開発をスタートし，結果は分散が非常に大きく，開発の 2 倍 3 倍というプロジェクトが存在し実績の平均値は計画から大きく乖離したものになる．プロセス実績分布の分散と平均値および計画との乖離の 3 点はプロセス実績から見ての成熟度評価の大切な要素となる．

一方，成熟した組織では現実に合わせた計画を立てるようになるために，未成熟な組織がたてる達成不可能な計画値より最初は悪い計画値になる．しかし，実績は計画に非常に近いものとなり結果的には未成熟な組織に比べて Q，C，D はずっと改善されたものになる．また，プロジェクトによるばらつきも少なくなり，プロジェクトが制御可能になって

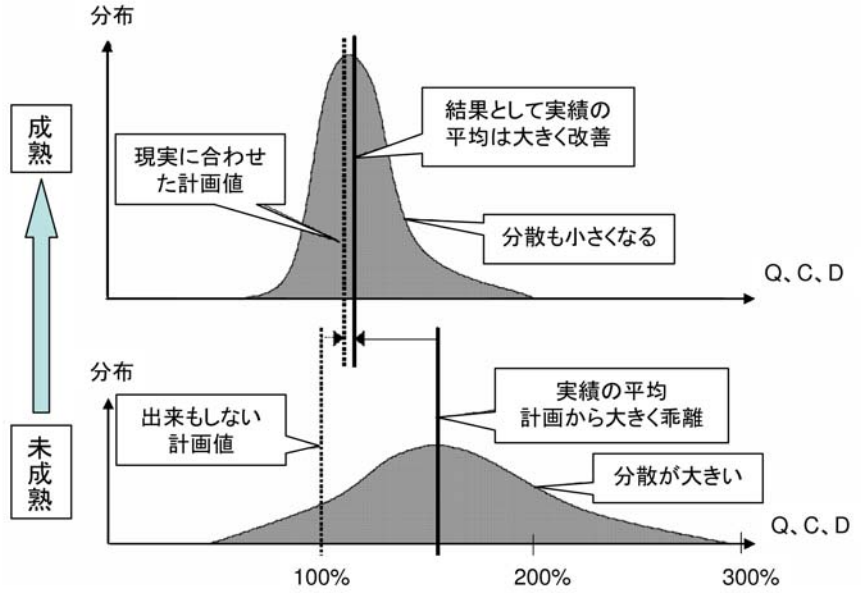


図 2.9: プロセス成熟度の向上と QCD の関係

いることを示している．

プロセス成熟度が上がっていくと，この分散と平均値および計画との乖離の 3 点が同時に良くなっていくと言われている．成熟度を上げていくためには階段を上げていく必要があり，途中で階段を省略することは出来ない．各階段から次の階段に上がるために特に注力すべき項目としてキープロセスエリアが定義されている．図 2.7 に 5 段階の成熟度レベルと各レベルに含まれるキープロセスエリアを示している．これらのキープロセスエリアはそれだけを行えば次のレベルに上がれるというものではなく，あくまでも各レベルにおける特に注意すべきポイントを述べている．

したがって各レベルのキープロセスエリアはそのレベルになって初めて実行するものではなく，それより以前から取り組んでおくべきものであるが，各レベルを達成してしっかりと歯止めをかけるためにはそれぞれのキープロセスエリアが必須であることを示している．

プロセス成熟度が上がっていくとその組織が実行するプロジェクトの可視性がどのように変化するかを示したのが図 2.10 である[3]．ソフトウェア開発は見えないとよく言われるがそれは成熟度レベルが低いからであって，成熟度が上がっていくとリアルタイムにプロジェクトの状態が数値で把握でき，プロジェクト管理へのフィードバックが可能となってくる．

典型的なレベル 1 の組織ではプロジェクトの開始と終了が分かるだけで，プロジェクトの実行中の状況は外部からは全くといってよいほど分からない．プロジェクトを実行して

[3]ソフトウェア能力成熟度モデル (TR24), 2.3 ソフトウェアプロセスの可視性, 図 2.3 より

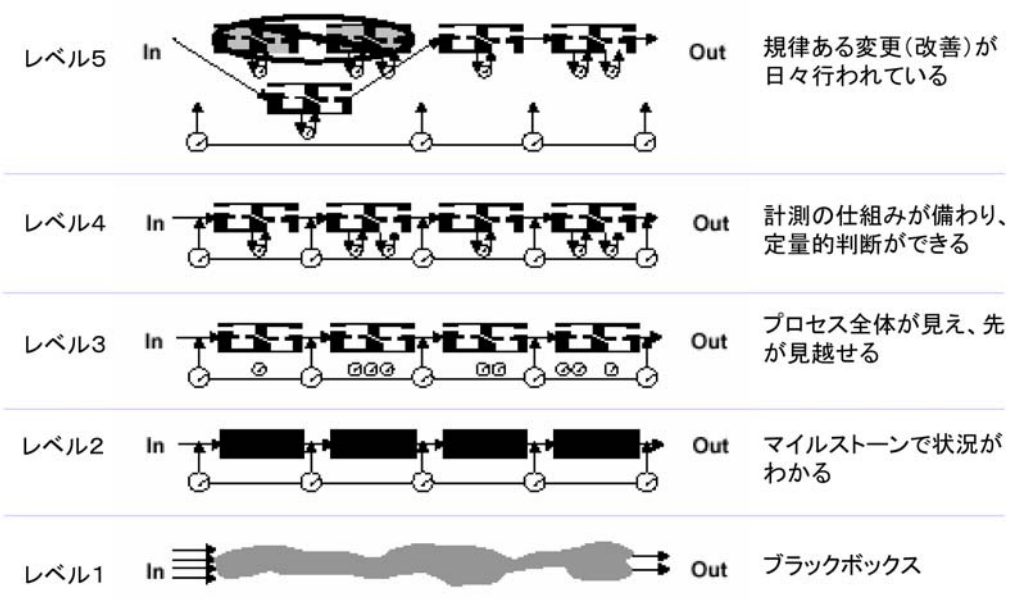


図 2.10: プロセス成熟度とプロジェクトの可視性

いる人達自身にも状況が分からないのがほとんである．これがレベル 2 になると工程単位程度のプロセスが定義され，この単位での Q，C，D の確認がされながらプロジェクトが実行される．レベル 3 になると各工程の作業レベルが定義され，この単位でプロジェクトの状況が定量的に把握できるようになる．レベル 4 になると各工程の作業レベルの進捗状況が定量的に把握され，当該作業制御のためのフィードバックとして使われる．さらにレベル 5 になるとプロセス定義も固定的なものではなくそれぞれプロジェクトで最適な状態にダイナミックに管理されるようになる [9].

2.7 CMM の活用状況と CMM に基づくプロセス改善の効果

CMU/SEI から定期的に発行されているレポート [39, Maturity Profile (September 2011)] によると，2006 年以降，約 5,000 の組織がアプレイザル[4]を行い，CMU/SEI に報告している．このデータは，全世界において広く CMMI が活用されていることを示している．また，このレポートの中では，成熟度レベルを一つ上げるための期間を以下のとおり報告している．

- 成熟度レベル 2 から 3：19ヶ月
- 成熟度レベル 3 から 4：21ヶ月
- 成熟度レベル 4 から 5：25.5ヶ月

[4]CMU/SEI から認定されたアプレイザによって，SCAMPI という手法に基づいて診断される．一般に，アプレイザルのために 5〜8 人程度のチームが編成され，2〜3 週間の期間をかけて実施する．

上記の結果は，組織においてプロセス改善活動を継続して実施することの必要性と成熟度をひとつ上げるためには 2 年前後の長い期間が必要となることを示している．このことから，2.4 節で示したとおり，プロセス改善活動を成功させ，成熟度を高めるためには，経営トップから改善推進者まですべての人の主体的な関与が必要であることが分かる．

次にプロセス改善の効果について説明する．プロセス改善の効果としては，一般的に以下のような効果があると報告されている．

- 生産性の向上
- テスト以前により多くの問題点を検出
- 市場投入までの時間の短縮
- 顧客先での欠陥の減少
- 要員の士気の向上
- 顧客満足の向上

定量的なデータの報告は少ないが，CMU/SEI より表 2.3 に示す CMM に基づくプロセス評価後の改善効果が報告されている [37]．それによると，プロジェクト失敗リスクの減少や，スケジュール遅れが原因での工数増大によるコストのオーバーランの減少などによって，生産性において年率で 61%程度の向上が期待できることが示されている．

表 2.3: プロセス改善の効果

分類	範囲	中央値	データ数
品質	48%	2〜132%	34
コスト	34%	3〜87%	29
生産性	61%	11〜329%	20
日程	50%	2〜95%	22
顧客満足	14%	-4〜55%	7
ROI(投資対効果)	4.0:1	1.7:1〜27.1:1	22

また，文献 [38, James Herbsleb] で，品質保証活動の徹底，要求仕様変更管理（構成管理）の徹底，レビューの徹底等の効果により，開発完了時点でソフトウェアに残存する欠陥の量は，年率で 40%近く減少することが報告されている．これは CMM 導入時におけるプロジェクト完了時点での平均欠陥残存率を，新規 KLOC[5] 当たりで 1 エラーであるとすれば，導入後 1 年で 0.6 エラーに低減させ，さらに次の年には 0.36 エラーにまで低減できる可能性があることを示している．また，納期の視点では，開発期間の短縮率で，1 年当たりで 20%前後の短縮が可能であることが報告されている．

生産性，品質，納期のすべての点で，上述のような改善が常に約束されるわけではないが，表 2.3 に示したように，CMM の導入にかかわる投資に対する効果を金額で換算した

[5] KLOC:KLine of Code の略．1KLOC はソースコード 1,000 行を意味する

比率は 5 倍程度で，投資対効果は十分であることが報告されている．

ただし，これらの事例での導入効果は，プロセス評価法としての CMM を導入したことが直接的な原因での効果ではなく，CMM によるプロセス評価を実施し，その評価を向上させるためのプロセス改善活動を実施した結果として得られた成果であることに注意しなければならない [41].

2.8 既往の研究

ソフトウェアプロセスの持続的な改善を推進する方法論として，能力成熟度モデル CMM[6]が多くの組織で活用されている．この CMM を利用し，SPI 活動をより効果的・効率的に推進するために，文献 [17, Fukuyama] では，推進体制や普及手段など 7 種類の広義の支援ツールが提案され，その支援ツールの適用による効果も示されている．図 2.11 に，これらの広義の支援ツールを組み上げ，プロセス改善活動を推進するためのシステムとして構築した例を示す.

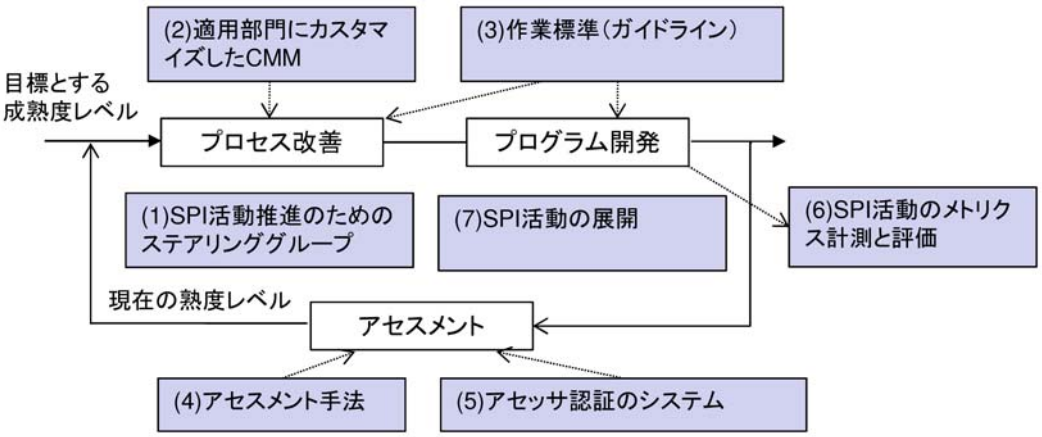


図 2.11: プロセス改善活動を推進するためのシステム

また，文献 [40, 板橋] では，PASSPORT（PAnasonic System and Software PrOcess impRovementT）アセスメント体系を構築し，あるべき姿としての CMMI とのギャップを見つけて改善するという取組みが紹介されている．

この PASSPORT アセスメントの特徴は，単に組織プロセスの評価を行うたけではなく，プロセス改善推進者の能力向上を図るために，全社組織横断でアセッサが参加する仕組みを構築したことである．例えば，正式な CMMI のアセスメントでは，アセスメントチームはリーダを除いて，アセッシ（アセスメント対象者）組織内の人間だけで構成するのが

[6]その進化版は，能力成熟度モデル統合 CMMI（Capability Maturity Model Integration）であり，2000 年半ば以降は，CMMI を利用することが一般的である．

一般的である．それに対し PASSPROT アセスメントでは，原則としてアセッシ組織とは異なる事業ドメインのアセッサによりチームが構成される．これによって，全社でのアセスメント評定の平準化や全社にまたがるベストプラクティスの共有が進められるというメリットが生まれる．

2.9 ソフトウェア開発組織におけるプロセス改善の難しさ

CMU/SEI は CMMI を活用するためのアプレイザルの実行プログラムも用意している．ただしこれはあくまでもプロセスの評価であって，図 2.9 に示したような Q，C，D プロセス実績の評価は含んでいない．CMMI はプロセスの改善点を発見しそれを改善していくためのモデルであるので，その結果である Q，C，D プロセス実績の評価は必要ではないという意見もあるが，その改善がどの程度有効であったかを評価し知識として蓄積するためにもプロセス実績の評価は必要であると思う．また CMMI は組織の成熟度を 1 から 5 の 5 レベルに判定するため，元々は良いプロセス実績を得るためのプロセス改善のはずが，高い成熟度の判定を得るためのプロセス改善に陥りやすい．CMMI はあくまでもモデルであり，すべての組織にそっくりそのまま当てはまるとは限らないので，各組織に適応した工夫が必要であるが，キープロセスエリアの表面的な項目のみを改善して高い成熟度レベル判定を得ても，良いプロセス実績が得られなければ無駄な努力をしているだけで，決してプロセス改善をしたとは言えない．アプレイザルとかレベル判定というと常にこのようなレッテルをもらうための活動となりやすいため，経営トップから改善推進者まですべての人が十分に心すべきところである [9].

また，CMMI などの改善モデルを使わずに改善活動を推進している組織もあるが（その多くの組織が課題解決型の改善活動を推進していると思われる），表 2.1 に示したとおり，“ 一過性になりやすい ”，“ 組織の力に制限された解決策の提示しかできない ” といった短所もあり，組織のプロセスを維持・改善するという活動を効果的に継続できないことも多い．

ソフトウェア開発の現場においては，図 2.1 で示したように,「設計品質の向上」,「開発コストの削減」,「開発期間の短縮」 といった課題が上位に挙がっている．どちらのタイプの改善活動であっても，これらの課題を直近のプロジェクトで解決することを優先するために，中長期的な視野を持って，2.4 節で示したような能力の高いメンバを SEPG に割り当てることが難しいという現実がある．

第3章　SPIフレームワークの提案

CMM が非常に注目を集めていた 1990 年代後半，SPI 活動を全社的に推進するという計画が示された．この時，一番重要なポイントとして考えたことは，いかに多くの開発部門にこの活動を展開し定着[1]させることができるのか，ということであった．

この活動の推進に割り当てられたメンバの多くは，1990 年代にいくつかの品質管理手法／品質管理ツールを社内展開した際，手法やツールの利用が組織横断的に定着する組織とそうでない組織があることを認識していた [24]．全社的に，SPI 活動を推進・定着させるための進め方を検討する際，この定着する組織の特徴を洗い出し，そこから得られた内容を分析し，SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」を導出した．

本章では，最初に，全社的な SPI 活動開始時の背景と SPI 活動の問題分析結果を示す．次に，プロセス改善活動が定着している組織の特徴を分析した結果を示し，その分析結果に基づいて導出した「SPI フレームワーク」を提案する．そして，その SPI フレームワークに対する要件を明確にする．さらに，この SPI フレームワークを東芝グループにおける SPI 活動の推進と定着に適用するために検討した内容と，その結果として構築した東芝版 SPI フレームワークの詳細について説明する．

3.1　全社的な SPI 活動開始時の背景

全社的に SPI 活動を普及・展開させるための活動は，2000 年に本社の技術企画室[2]の主導で立ち上がった「設計力強化プロジェクト」のワーキンググループの一つである「組織力強化ワーキング」が出発点である．

1990 年代後半，ソフトウェア／ハードウェアの開発量が大幅に増大したことで，以下に示すことの実現が強く求められていた．

- 開発量の増大に対応できる設計効率の実現
- 設計上流での設計品質の向上
- 設計期間の短縮と適切な市場投入のタイミング

[1]本論文では，開発部門の中で定常的に SPI 活動が実施されている状態を定着と捉えている
[2]東芝グループ全体の技術行政を司る組織

上記を実現するためのポイントは設計力である．設計力を強化するためには，技術的側面だけではなく，組織的側面からの改革も必要になってくる．そこで，ワーキングの一つとして「組織力強化ワーキング」が設置された．さらに，組織力を強化するための技術として CMM を利用することになった．

この当時，海外企業への製品納入に際して，CMM で示されているレベルを受注要件として求める企業もあった．また，日本国内においては，政府調達の重要な条件として CMM によるレベル認定が求められるという動きもあった [42]．このように，日本においては CMM が非常に大きな話題となっていた．しかしながら，CMM ベースによるプロセス改善活動については，2.9 節で示したような“ レベル取得 ”に陥ってしまい本質的な改善につながらない場合も多いとの弊害も認識されていた．

このような状況のもと，組織力強化ワーキングでは，多くの開発部門が CMM をプロセス改善活動に活用することで，以下を実現することを目的として活動を開始した．

- **組織力強化ワーキングにおける活動の目的**
  - ソフト生産技術力の向上
  - 改善のできる組織文化の構築

このプロジェクトには，最初の 2 年間は本社から活動資金が提供された．その後は，全社のコーポレート部門（ソフトウェア技術センター）が継続して組織力強化のための活動を推進するという方針で進められた．

3.2　プロセス改善活動の問題分析

多くのソフトウェア開発部門では，2.1 節で示したような問題を解決するため，組織的にさまざまな改善活動を推進している．また，企業としても，いろいろな観点からの活動や運動が展開されている．企業や組織においては，経営者やリーダはある一定の期間で交代となる．新しくそのポジションに付いた方は，当然，自分なりのやりたいことがあり，何かを変える（その呼び方は改善，改革，運動などさまざまである）ことを考える．このような活動は当然必要であるが，継続性という観点からは場当たり的な印象を与えることも多い．

一方，組織の開発プロセス整備という観点からすると，改善活動に対するリソースや優先順位といった問題がある．本来，ソフトウェア開発部門においては，プロセスオーナーシップを持ち，自分たちのプロセスは自分たちで作り，改善するというサイクルを回すことが求められている．それが，製品ソフトウェアを開発するプロフェッショナルとして要求されていることでもある．しかしながら，開発規模の増大，開発期間の短縮など，非常

に厳しい開発状況の中では，十分なリソースを改善活動に割り当てられなかったり，開発の忙しさの中で，改善活動の優先順位が下がったりしてしまうこともある．

さらに，改善活動の進め方に対する技術やノウハウがない，新しい開発技術や管理技術に解決を求めてしまいがちといった状況もよく見受けられる．また，他部門の良い取り組み事例や失敗事例を参考にしたいと考えても，組織間の交流が十分でない場合が多いため必要な情報が得られず，結果的に，自分たちだけの閉じた世界で悩みを抱えてしまうということもよくある状況である．

このような問題があり，数々の技術施策を実施するものの，効果的・効率的なプロセス改善につながらない場合が多いと考えられる．

結果的に，組織内にプロセス改善活動の成果を展開できないという現実と，少ないリソースで広い範囲をカバーせよという要求にはさまって，プロセス改善を推進する担当者のモチベーションも下がり，効果的な活動ができず，組織的に認知されないで活動が終わってしまうことが起こり得る．このような活動が立ち上がっては消える，ということを繰り返すため，改善活動を継続的に実施するという体制と組織文化が確立できないことが多い．

この問題を解決するためには，各組織において“「改善の責任」と「改善技術／改善ノウハウ」を継続的に担う”ことのできるメンバを割り当て[3]，そのメンバが中心となって改善活動を推進する体制を構築することが必要である．

3.3 全社的な SPI 活動開始時の留意点

SPI 活動を全社的に推進するという計画において設置された組織力強化ワーキングを立ち上げる際に一番重要なポイントとして考えたことは，大規模組織[4]において，いかに多くの開発部門にこの活動を展開し定着させることができるのか，ということであった．これは，3.1 節で示した本活動の目的「改善のできる組織文化の構築」のひとつの成果指標でもある．

図 3.1 に，東芝グループの組織体制を示す（図中に示したカンパニー名称は，2011 年 10 月時点のものである）．

図 3.1 に示したとおり，東芝グループは，コーポレートとカンパニーから構成されている．コーポレートは，スタフ部門，研究開発部門などから構成されている．カンパニーは，事業領域毎に設置される．カンパニーは，複数の製品開発部門とグループ会社から構成される．また，グループ会社も複数の製品開発部門を持つ．

[3]ソフトウェア開発部門においては，多くの場合，開発プロジェクトと兼務でこのような役割を持たせることが多いと思われる．メンバが固定してしまうのも弊害があるため，組織内でのローテーションなどを工夫する必要がある．

[4]本論文では，おおよそ 50 以上の開発部門を持つ組織を大規模組織と考えている

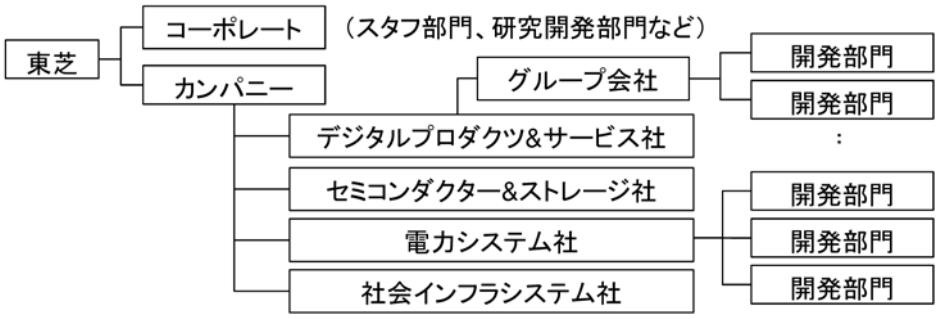


図 3.1: 東芝グループの組織体制

3.2 節の問題点をまとめると，プロセス改善活動を成功に導くためには，以下の 3 つの観点からの取り組みが必要と考えた．

1. プロセス改善の推進体制
   ソフトウェアを開発している組織の文化を変えようとした場合，場あたり的なアドホックな体制でプロセス改善を試みても決して有効に機能しない．巨大化した組織を適切に運営するために階層化されたマネジメント体制が有効に機能するが，これと同様にプロセス改善の推進についても，その推進組織を有機的に構築し，階層的な役割分担によって進める体制を構築する必要がある．

2. プロセス改善推進のためのインフラ構築
   たとえ上述のようなプロセス改善のための推進体制が確立できたとしても，この組織を支えるためのインフラが整備されていなければ，その組織は効率的な活動ができない．特に，有機的に編成されたプロセス改善組織を，ある意思をもって一体として動かすためには，情報面でのインフラ整備が重要な要因となる．

3. 意識統一と改善のスキル向上
   プロセス改善を組織的に進めるためには，改善対象であるソフトウェア開発部門全てのメンバがプロセス改善に対して正しい認識を共有することが必須であり，このことがプロセス改善の推進をより円滑にかつ強固にする．またプロセス改善を推進するメンバのスキルを向上させることで，プロセス改善活動はより有効に機能する．このためにはプロセス改善に関する教育コースなどによる技術レベルの底上げが必要である．

3.4　定着できる組織の特徴と SPI フレームワークの提案

SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」を導出するために，過去の経験に基づき，手法やツールの導入が定着し，継続して活用されている組織の特徴を分析した．定着する組織の分析結果を図 3.2 に示す．

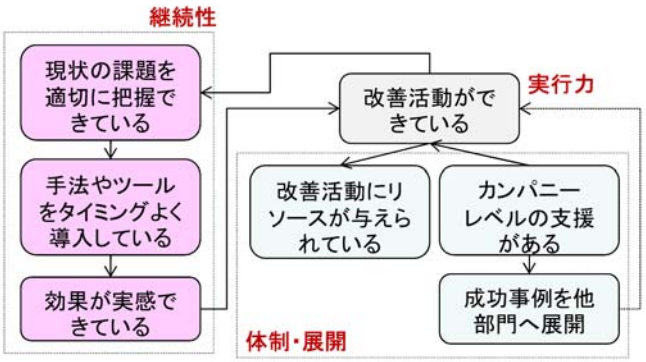


図 3.2: 継続できる組織の特徴

手法やツール利用が組織横断的に展開できている組織の特徴として，最も重要なポイントは，当然のことであるが改善活動ができているということであった（実行力）．この実行力を支えるものが体制・展開という観点である．さらに，実行力を支えるために，継続性という観点が重要であることが分かった．

次に，図 3.2 で示した特徴を持つ組織を構築するために求められる能力や活動を洗い出した．その結果を，図 3.3 に示す．

この結果から，開発部門がここで示した能力を保持するか，あるいはそれと同等のものが提供され，活動が実践されれば，今まで改善活動が継続できなかった組織でも，継続できる可能性が高くなると考えた．そこで，抽出できた能力と活動を，SPI フレームワークを構成する要素として捉えて，それぞれの構成要素を有機的に連携して機能させる枠組を明確にした．図 3.4 に，継続できる組織の特徴を分析した結果から導出した SPI フレームワークを示す．

図中の活動 1～活動 6 は，次節で説明する SPI フレームワークに求められる要件との関係を示すために付加したものである．提案する SPI フレームワークは，6 つの要素から構成される．フレームワークの中心は,「推進体制を構築して改善活動を実践する」ことである．さらに，改善活動をより良く実践するために,「改善活動を実施する能力」,「改善活動の進め方に関する能力」,「管理手法／管理ツールを調査して選定する能力」を利用する．さらに，改善活動の実践結果を提供することで，それが資産として蓄積され，他部門の改善事例を利用できるようになる．また，改善活動の成果を論理的に提示することで，改善

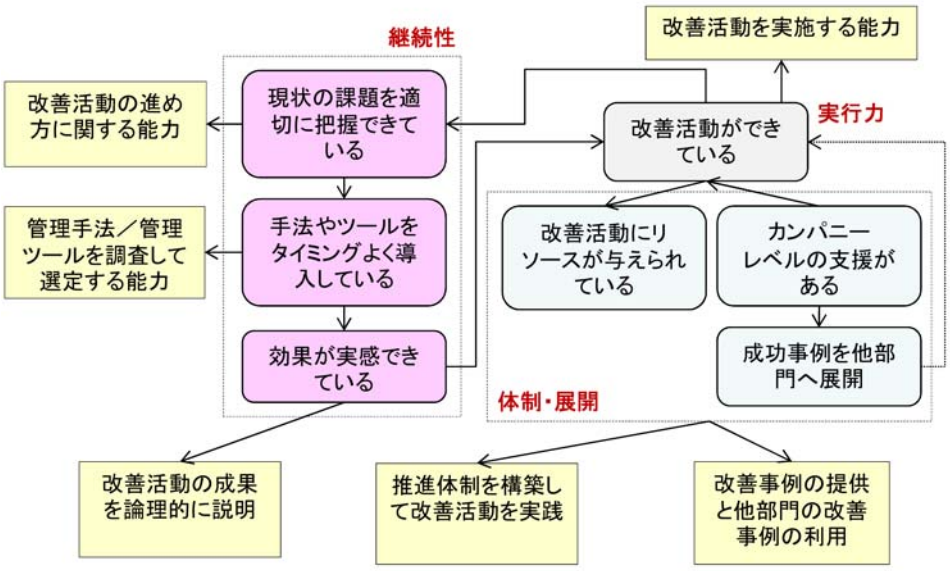


図 3.3: 改善活動を継続するために求められる能力と活動

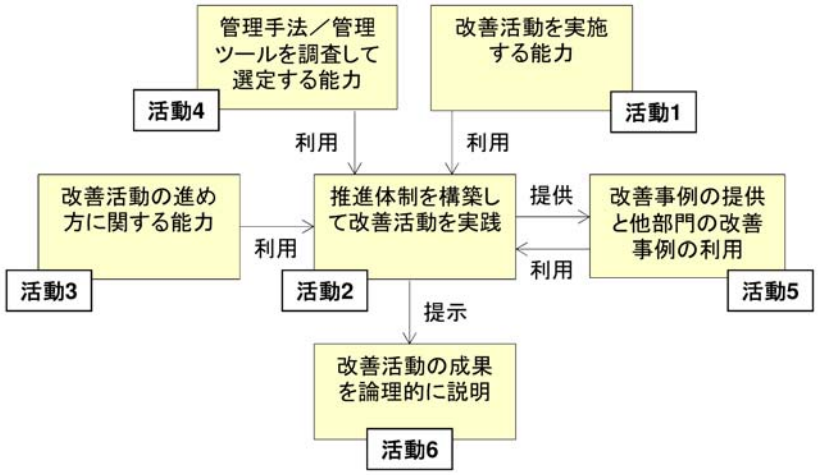


図 3.4: SPI フレームワーク

活動に対するスポンサーや組織からのコミットメントが獲得できる.

3.3 節で示した本活動開始時の留意点と SPI フレームワークを構成する要素との関係を表 3.1 に示す.

表 3.1 の結果から，大規模組織において全社的な SPI 活動を展開する際の留意点として挙げた項目が，提案した SPI フレームワークではカバーされていることが確認できた．この提案は,「組織力強化ワーキング」の上位組織に提案され承認されたことで，具体的にこの SPI フレームワークに基づいた東芝版 SPI フレームワークの構築作業が始まった.

表 3.1: 活動開始時の留意点と SPI フレームワークとの関係

活動開始時の留意点	SPI フレームワークで対応する構成要素
プロセス改善の 推進体制	活動 2：推進体制を構築して改善活動を実践 活動 6：改善活動の成果を論理的に説明
プロセス改善推進 のためのインフラ構築	活動 4：管理手法／管理ツールを調査して選定する能力 活動 5：改善事例の提供と他部門の改善事例の利用
意識統一と改善の スキル向上	活動 1：改善活動を実施する能力 活動 3：改善活動の進め方に関する能力

3.5 SPI フレームワークに対する要件

全社的な SPI 活動の推進・定着を促進する仕組みである「SPI フレームワーク」は，前節で示したとおり 6 つの要素から構成されている．

一般に，大規模組織では，多種多様な製品やサービスを提供している．当然，製品やサービスを提供するための開発プロセスや品質保証の仕組みも違ってくる．また，個々の事業を司る組織の文化はそれぞれ異なる．そのため，製品，市場，事業の特性を考慮した改善活動を推進しなければならない．

このような大規模組織における特性を考慮したうえで，SPI フレームワークを構成する各要素に対する要件を以下のとおり定義した．

1. 活動 1：改善活動を実施する能力

   組織の成熟度を高めるためには，中長期的な視点からのあるべき姿を持つことが大切である．また，そのあるべき姿に到達するためには，改善のサイクルを回し続けなければならない．企業全体で SPI 活動を推進するためには，中長期的な視点からのロードマップを示してくれるモデルの選定と，改善活動を効果的・効率的に回し続けるための方法論が必要である．

2. 活動 2：推進体制を構築して改善活動を実践

   各開発部門に SEPG を設置して，全社的に SPI 活動を展開するためには，各開発部門における SPI 活動に対して直接的，間接的な支援を求められることが多い．大規模組織の中で複数の開発部門における SPI 活動を支援するためには，必然的に人的リソースが不可欠である．また，製品，市場，事業などの特性を考慮した適切な SPI 活動を推進するためには，推進部門と開発部門との間をつなぐ役割を持ったメンバを含めることが必要である．

3. 活動 3：改善活動の進め方に関する能力

   製品の多種多様化に伴い，組織を横断したクロスファンクションチームを作り製品

開発を進める機会が多くなってきている．開発プロセスの共通概念を持ち，改善活動に対して共通の理解を持つことは，プロジェクトを成功につなげるひとつの重要な要素である．したがって，開発プロセスの基本と改善活動の進め方の習得は必須である．また，開発部門間の SPI 活動推進に関する能力にバラツキがでないように，企業全体での底上げを考慮しなければならない．

4. 活動 4：管理手法／管理ツールを調査して選定する能力
   開発部門では，製品開発に組織のリソースの多くを使うため，管理手法や管理ツールの有効性が理解できていても，導入・評価し，組織内に定着させるまでのステップを完遂させることが難しい．管理手法や管理ツールを実際の製品開発で使えるレベルのものに仕上げることと，それを開発部門内に展開するための取り組みが必要である．

5. 活動 5：改善事例の提供と他部門の改善事例の利用
   開発部門での SPI 活動は，自部門に閉じた活動になりがちである．したがって，さまざまな解決方法の中から，適切なものを選択できるように，他部門の活動状況や開発プロセス，ベストプラクティスなどを参照できる情報基盤の整備が必要である．また，SPI 活動は成果が見えづらいという側面があるため，推進担当者や開発部門のメンバのモチベーションを維持することが大きな課題である．改善活動の悩みを相談し解決のためのヒントを得るための場や人的ネットワークを提供することは，モチベーションの維持に不可欠である．

6. 活動 6：改善活動の成果を論理的に説明
   SPI 活動は，投資に対する効果がどちらかというと間接的・中長期的であることが多い．一方で，SPI 活動の投資対効果を経営層や管理者層に示すことが，コミットメントを維持するために重要である．したがって，活動の成果（プロセスの実施・成熟度の向上）と活動の効果（品質・納期・コストの向上）を論理的に関係付けて捉え，可視化することが必要である．

3.6 SPI フレームワークを構成する各要素の実装結果

前節で示した SPI フレームワークを構成する各要素に対する要件に対して，東芝グループ全体で SPI 活動を推進することを考えた場合，どのような内容にすればよいのかを検討した．ここでは，その検討結果と具体的に実装した内容について説明する．

3.6.1 活動 1：改善活動を実施する能力

改善のモデルとして CMM（途中から CMMI）を選択した．また，SPI 活動における PDCA（Plan-Do-Check-Act）のサイクルとしては，IDEAL[5]を利用することにした [18]. CMM，IDEAL ともに，ただ単にモデルをそのまま利用するのではなく，東芝グループの事例を交えた以下のガイドを作成した．

- 全てのプロセス領域に対応したガイド．CMM では 19 個のプロセスに対応したガイドを作成．CMMI では 24 個のプロセスに対応したガイドを作成．
- IDEAL で示されている 5 つのフェーズでの留意点やポイント，失敗事例などを示したガイドを作成．

CMM における要件管理プロセスのガイドの目次ページを図 3.5 に示す．その他のプロセス領域のガイドも，目次構成は共通である[6]．それぞれのガイドは，50〜70 ページである．

IDEAL ガイドでは，5 つのフェーズ毎に，目的，開始基準／完了基準，活動，具体的な活動，留意事項，ケーススタディの 6 項目を説明している．具体的な活動，留意事項には，過去経験してきた SPI 活動の実践経験から得られたノウハウやポイントを盛り込んでいる．このガイドは約 130 ページである．また，IDEAL の手順にしたがってプロセス改善活動の計画を立案しやすくするために，IDEAL 白地図と呼んでいるテンプレートも作成した．

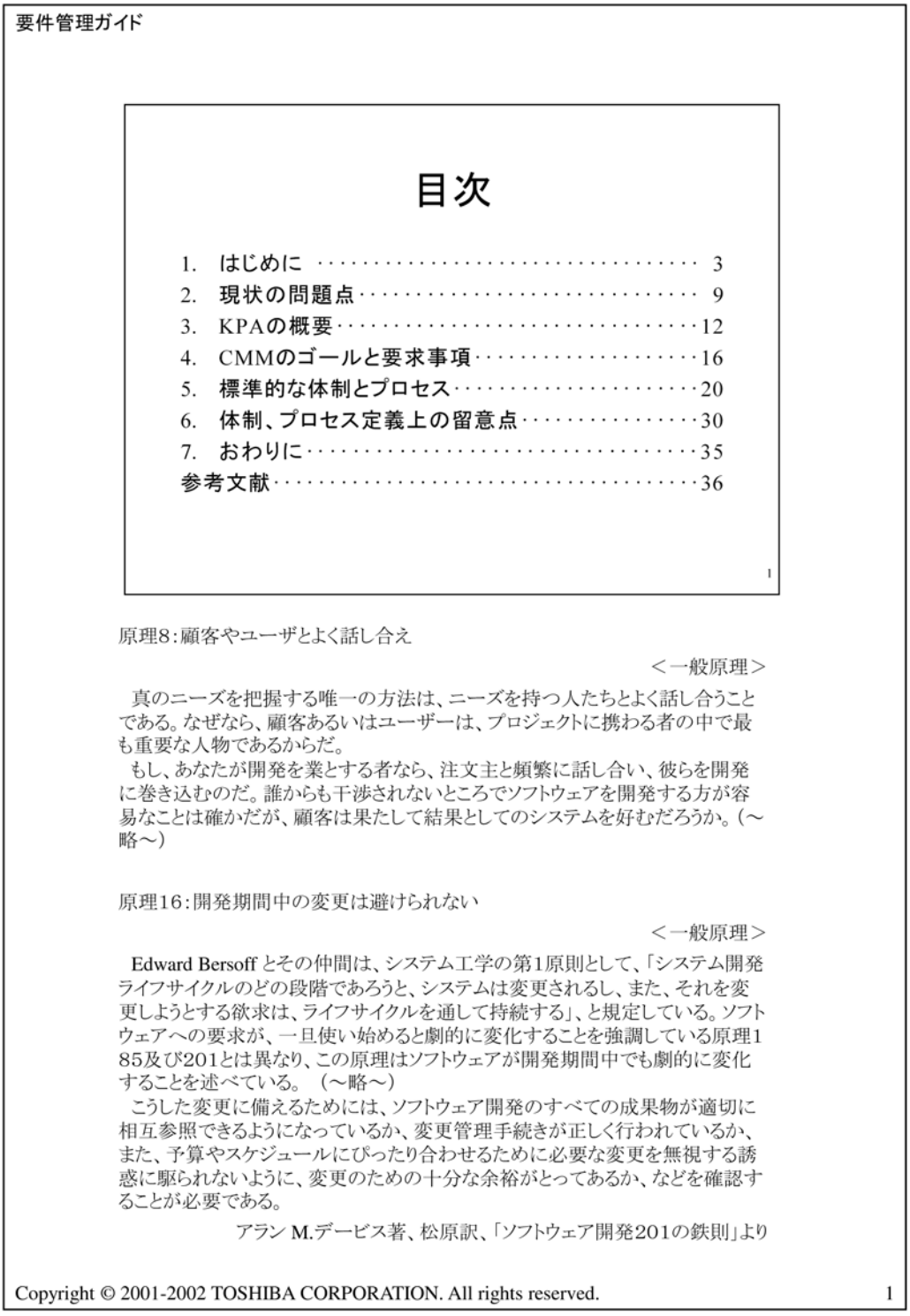
3.6.2 活動 2：推進体制を構築して改善活動を実践

大規模組織における事業分野は幅広く，それぞれの事業を統括する組織の文化も異なる．開発部門の状況に合わせたきめ細かい支援を実施するには，事業分野毎に SPI 活動を支援する部門を設置することにした．また，SPI 先端技術の研究・開発を加速し，SPI 活動のノウハウやベストプラクティスの共有などによるシナジー効果を創出し，全社的に SPI 活動を推進していくためには，全社レベルで SPI 活動をとりまとめる SPI 推進部門を確立することが必要と考えた．このような分析の結果，開発部門，事業部門，全社の 3 階層で SEPG 体制を構築し，SPI 活動を推進することにした．それぞれの階層における名称を，開発部門 SEPG，カンパニー SEPG，コーポレート SEPG とした．構築した 3 階層 SEPG の概要を，図 3.6 に示す．

図 3.6 に示したとおり，開発部門における改善活動は，開発部門 SEPG が主体となって進める．ソフトウェア開発部門の SPI 活動を，カンパニー SEPG とコーポレート SEPG

[5]Initiating(開始)，Diagnosing(診断)，Establishing(確立)，Acting(実践)，Learning(学習) の 5 つのフェーズから構成されている．

[6]目次のノートのところには，“アラン M. デービス著，松原友夫訳「ソフトウェア開発 201 の鉄則」，日経 BP 社，1996 ”から関連する原理を選択して載せている．


要件管理ガイド

目次

1

原理8：顧客やユーザとよく話し合え

＜一般原理＞

真のニーズを把握する唯一の方法は、ニーズを持つ人たちとよく話し合うことである。なぜなら、顧客あるいはユーザーは、プロジェクトに携わる者の中で最も重要な人物であるからだ。

もし、あなたが開発を業とする者なら、注文主と頻繁に話し合い、彼らを開発に巻き込むのだ。誰からも干渉されないところでソフトウェアを開発する方が容易なことは確かだが、顧客は果たして結果としてのシステムを好むだろうか。（～略～）

原理16：開発期間中の変更は避けられない

＜一般原理＞

Edward Bersoff とその仲間は、システム工学の第1原則として、「システム開発ライフサイクルのどの段階であろうと、システムは変更されるし、また、それを変更しようとする欲求は、ライフサイクルを通して持続する」、と規定している。ソフトウェアへの要求が、一旦使い始めると劇的に変化することを強調している原理185及び201とは異なり、この原理はソフトウェアが開発期間中でも劇的に変化することを述べている。（～略～）

こうした変更に備えるためには、ソフトウェア開発のすべての成果物が適切に相互参照できるようになっているか、変更管理手続きが正しく行われているか、また、予算やスケジュールにぴったり合わせるために必要な変更を無視する誘惑に駆られないように、変更のための十分な余裕がとってあるか、などを確認することが必要である。

アラン M.デービス著、松原訳、「ソフトウェア開発201の鉄則」より

1

図 3.5: CMM「要件管理」プロセスの目次ページ

が支援する．カンパニー SEPG は，事業所密着型技術支援部隊として位置付けられる場合もあれば，推進戦略と技術サポートを分離して組織を構築している場合もある．また，カンパニーレベルで SEPG を設置できない場合には，コーポレート SEPG がカンパニー

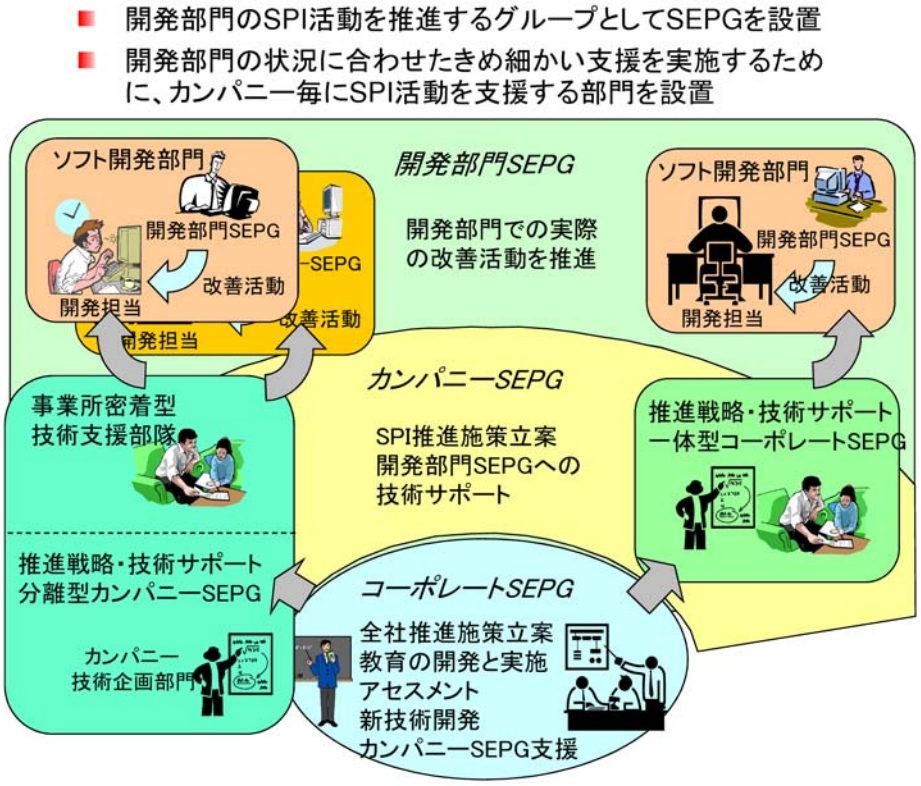


図 3.6: 3 階層 SEPG の概要

SEPG の役割を担い，開発部門の SPI 活動を支援することもある．

各 SEPG の役割と責任を表 3.2 に示す．

表 3.2: 活動開始時の留意点と SPI フレームワークとの関係

SEPG の分類	役割と責任
開発部門 SEPG	・自部門の SPI 推進施策の策定 ・自部門の SPI 活動の推進 －カンパニー SEPG，コーポレート SEPG で開発した手法やツールを利用して，SPI 活動を推進
カンパニー SEPG	・カンパニーの SPI 推進施策の策定 ・開発部門 SEPG の支援 － SEPG 会議へ参加（コンサルティング） －プロセスの開発・試行・運用に関する支援 －開発部門 SEPG や開発担当者へのトレーニング ・カンパニーで共通に利用できるプロセス，資産，インフラ（ツール，サーバなど）の整備，運用
コーポレート SEPG	・コーポレートの SPI 推進施策の策定 ・開発部門 SEPG，カンパニー SEPG の支援 ・トレーニング教材開発・トレーニング実施 ・プロセス診断の実施 ・情報共有機会の提供 ・SPI に関連する先端技術の研究・開発

開発部門における SPI 活動の推進においては，SEPG と SQAG[7]の果たす役割が重要である．SEPG は改善活動に責任を持つグループであり，組織の改善活動を推進する．SQAG は，組織やプロジェクトの活動が，決められたプロセスに遵守しているかどうかを確認するとともに，プロセスをより良く活用するためのサポートをすることが主な役割である．

ソフトウェア開発プロジェクト，SEPG，SQAG の関係を図 3.7 に示す．全社的な SPI 活動推進においては，開発部門毎に SEPG と SQAG の役割を持つグループを設置することを基本と定めた．

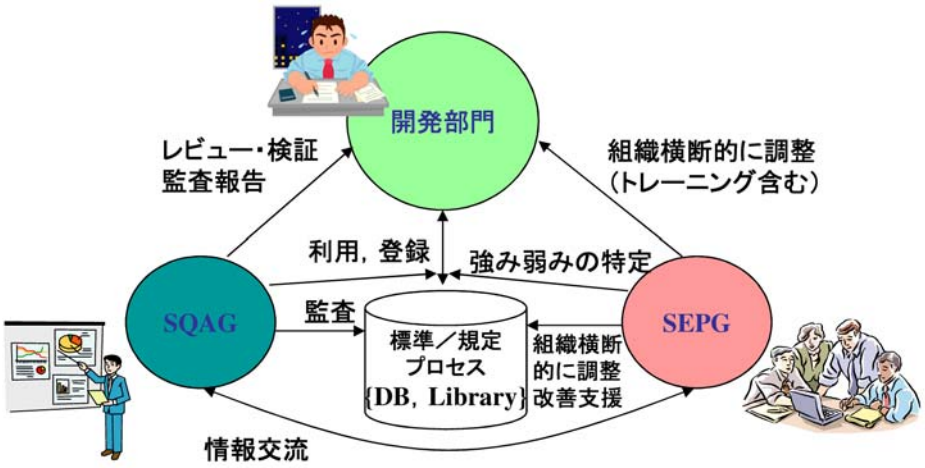


図 3.7: SEGP と SQAG のプロジェクトや組織に対する役割

3.6.3 活動 3：改善活動の進め方に関する能力

全社的な SPI 活動を開始する際に，SEPG メンバおよび SQAG メンバのためのトレーニング体系が必要と考え，図 3.8 に示すトレーニングコースを提供することにした．

SPI 活動を効果的・効率的に実践するには，開発部門およびカンパニーで SPI 活動を推進する SEPG リーダの役割が非常に大きい．ETSS(Embedded Technology Skill Standards) では，プロセス改善スペシャリストに関する技術体系が提示されているが [49]，どのような内容で，どの程度の時間をかけたカリキュラムを準備すればよいかは明確になっていない．そこで，ETSS などのスキル標準を参考にして，SPI 技術として習得すべき項目を明確にしたうえで SEPG リーダコースを開発した[8][59][51]．図 3.9 に，SEPG リーダコースの概要を示す．このリーダコースは，約 11 日間（各月 2〜4 日，4ヶ月）のトレーニングであり，CMMI と IDEAL を十分に理解してもらい，実践の場で活用できる能力を身に着

[7]Software Quality Assurance Group の略．SEPG が中心になって整備したプロセスの遵守状況やプロジェクトの成果物の品質チェックなどが主な役割

[8]2002 年に開発した当時は，ETSS などの標準スキルは存在しなかったため過去の経験をベースにカリキュラムを決めた．改訂時に ETSS などの標準スキルを参考にしている

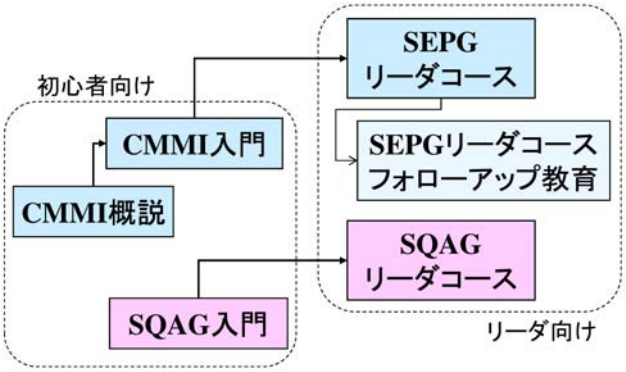


図 3.8: SEGP と SQAG のためのトレーニング体系

けてもらうことが主目的である．カリキュラムとしては，成熟度モデル，プロセス改善活動の進め方，一般スキル／関連技術，情報共有と知識共有というグループ毎に，必要なモジュールを定義している．さらに，SPI 技術を自分の活動に照らし合わせて習得できるように，IDEAL のステップに沿って SPI 活動を実践してもらい，最後に発表の場を設けている．

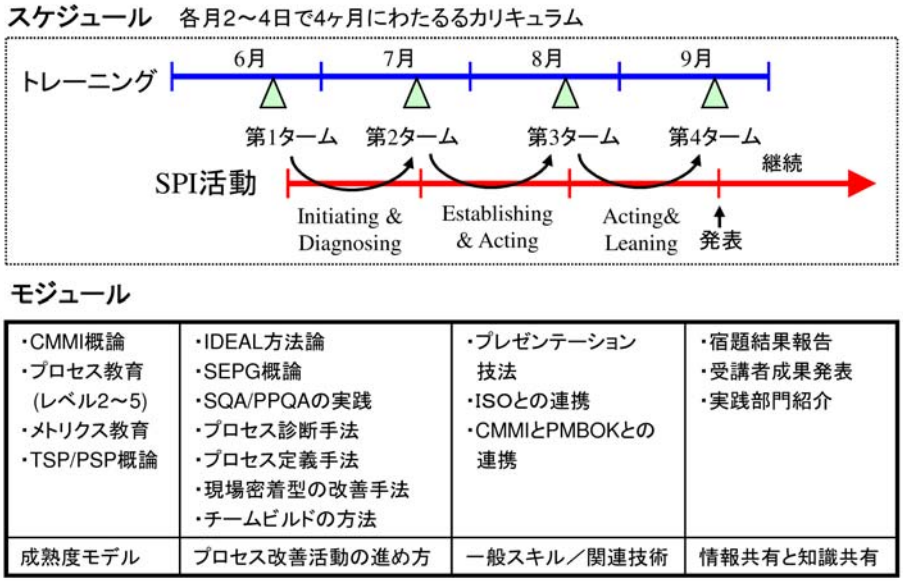


図 3.9: SEPG リーダコースの概要

SQAG トレーニングの全体像を，図 3.10 に示す [52].

トレーニング以外には，コンサルティング活動で利用する道具として，CMMI ガイドブック，推奨プロセスなどを開発し提供している．プロセス診断は，開発部門における開発プロセスの状況や SPI 活動の結果を評価するひとつの方法として利用されている．この

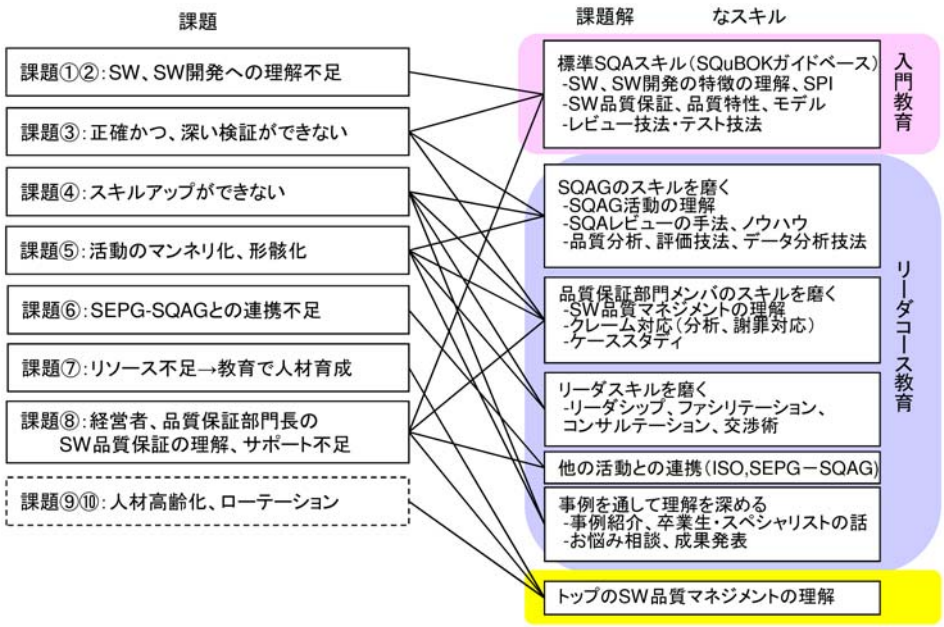


図 3.10: SQAG トレーニングの全体像

プロセス診断は，CMU/SEI から認定を受けたアプレイザーが行うものと，社内で認定されたアセッサが行う 2 つの方法を社内公式アセスメントとして提供している．開発部門は，プロセス診断の目的に応じて，使い分けることができる．

3.6.4 活動 4：管理手法／管理ツールを調査して選定する能力

開発部門において，SPI 活動の推進者が一番悩むことは，”具体的に何をどう改善したらよいか分からない”，”どこから着手したら良いのか？”といった，SPI 活動初期の立ち上げである．また，組織のマネージャや担当者からは，”何か効果的な方法やツールはないのか？”と求められることもよくある．そこで，SPI 活動をスムーズにスタートさせるために，あるひとつの具体的な改善活動の導入から定着までをサポートするための管理手法や管理ツールを改善ソリューション[9]として提供することにした．

選定した管理手法と管理ツール

図 3.11 に，1990 年代以降のソフトウェア品質管理技術に関する主要なトピックスと開発部門で導入の要望が多かった技術を示す．

この中から，開発部門からの導入の要望が多くあった以下の管理手法と管理ツールを提供できるように整備し，改善ソリューションとして提供している．

[9]改善ソリューションとは，特定の問題領域に対して改善活動を効果的・効率的に実践するために，ツールや技法が整備され，導入ステップが定められているものを示す．

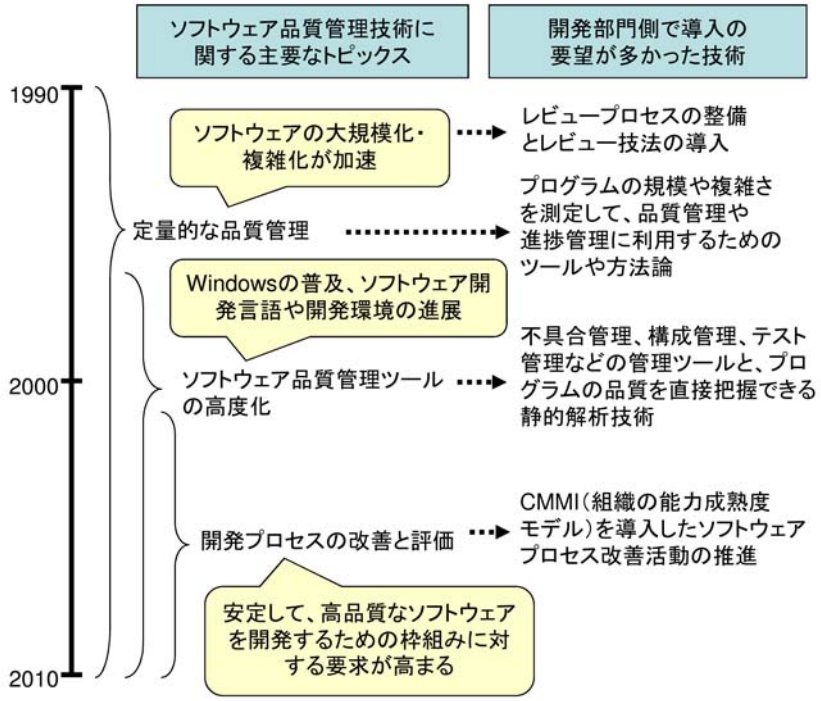


図 3.11: 管理技術の主要なトピックと開発部門からの要望

- メトリクスを活用した品質管理手法
- レビュー技術を用いた品質向上手法
- プログラム静的解析ツール
- 不具合管理ツール
- テスト管理ツール

全社展開を可能とするまでのステップ

選定した管理手法や管理ツールを，組織内に展開するためには，手法やツールの開発だけではなく，それらの運用方法を提示し，導入の手順などを整備する必要がある．そこで，管理手法や管理ツールを全社に展開する際には，図 3.12 で示したステップで準備を進め，改善ソリューションとして仕上げることにした．

図 3.12 で示した 3 つのステップの概要を以下に示す．

1. ステップ 1：手法／ツールの分析や開発
   対象とする手法やツールを有効に活用するために必要なデータ分析方法の検討や機能開発を行う．例えば，メトリクスを利用した品質管理手法の導入を考えた場合，メトリクスを計測するツールの開発や選定は当然必要であるが，収集されたメトリクスに対する分析方法を提示することも必要である．

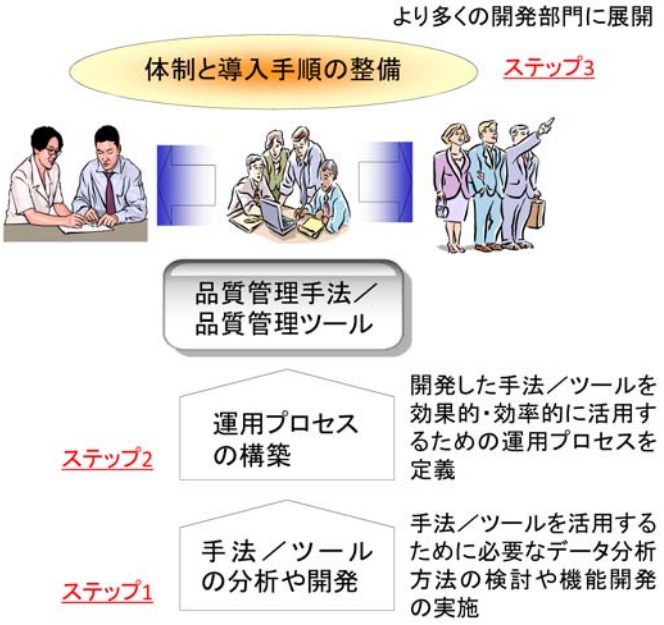


図 3.12: 改善ソリューションとして仕上げるまでのステップ

2. ステップ 2：運用プロセスの構築
   開発した手法／ツールを効果的・効率的に活用するための基本的な運用プロセスを定義する．

3. ステップ 3：体制と導入手順の整備
   多くの開発部門に展開するためには，推進体制を確立するとともに，導入作業を効率化することと，導入支援作業の負荷をうまく分散する仕組みが必要である．

上記ステップ 1 の取り組み事例として，付録 A に「メトリクスを活用した品質管理手法」，付録 B に「プログラム静的解析技術の効果的な活用方法」の研究・開発結果を示す．

3.6.5 活動 5：改善事例の提供と他部門の改善事例の利用

ベストプラクティスとして他部門の改善事例を流通させるとともに，SPI 活動に参画する開発部門や個人の質的・量的拡大をはかり，SPI 活動を推進しているメンバのモチベーションを維持・向上させることを目的としてイベントを開催している．

図 3.13 に，コーポレート SEPG が提供しているイベントの種類と個々のイベントで想定している参加者層を示す．全社に向けて SPI 活動を継続的にサポートし続けるというメッセージを出すという意味合いも込めて，これらの活動は定期的に，毎年決まった時期に開催している．

上記に示したイベントも含め，以下の 3 つの情報共有のための仕組みを整備した．

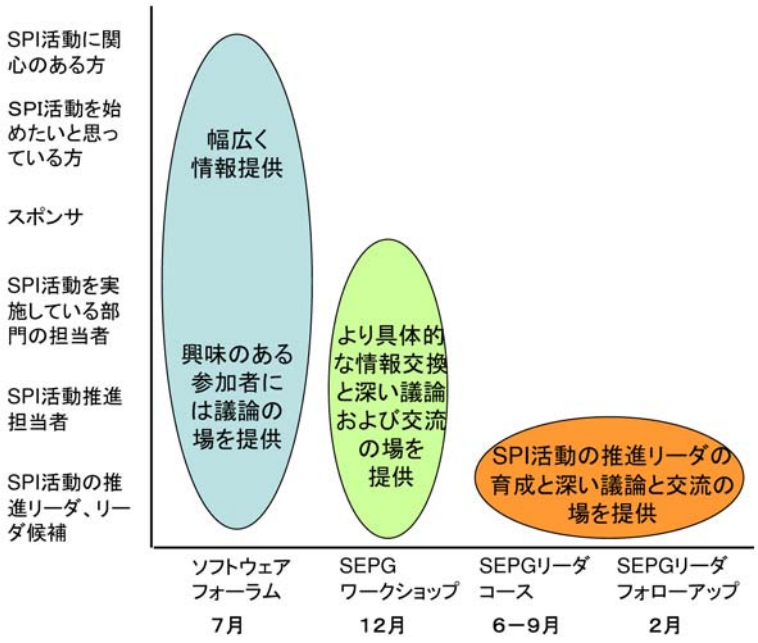


図 3.13: イベントの種類と想定している参加者層

- イベント：社内外の状況を知って刺激を受けるとともに，新たな取り組みのヒントを発見する場を提供する．
- 情報発信：SPI 活動に関する情報をタイムリーに発信する．
- 推進施策検討 : 全社的な SPI 活動推進状況の確認と，推進施策を検討する．また，カンパニー SEPG 間の情報共有の場としても活用する．

3.6.6 活動 6：改善活動の成果を論理的に説明

SPI 活動の実績から，SPI 活動の成果と効果を定期的に報告する仕組みを確立することは，SPI 活動に対する管理者層のコミットメントや開発担当者のモチベーションを維持するうえで非常に重要である．

そこで，SPI 活動の成果と効果を示すため，開発部門 SEPG から SEPG と SQAG の活動工数，プロセス領域毎のプロセス定着度合，プロジェクトの期間・工数・不具合の予定と実績などのデータを提供してもらい，データベース化している．開発部門 SEPG から提供されているデータを表 3.3 に示す．

データベースに蓄積された結果を分析し，全社レベルでの活動状況をまとめた SPI 活動レポートと個別フィードバックレポートを発行している．図 3.14 に，データ収集から結果の提示までの手順を示す．

このレポートの中で，SPI 活動の成果は，定着状況や組織成熟度の達成状況を使って説

表 3.3: 開発部門 SEPG から提供されるデータ

項目		データ
対象組織に関する内容		・製品ドメイン，ソフトウェア技術者数など
定着状況	SEPG定着状況	・SEPG人数
		・SEPG活動工数
	SQAG定着状況	・SQAG人数
		・SQAG活動工数
定着成果	プロセスの実装/改定	・KPA毎(PA毎)の実装数
		・KPA毎(PA毎)の実装数
	QCDの予定と実績	・Q:不具合，後戻り工数の予定と実績
		・C:開発工数の予定と実績
		・D:開発期間の予定と実績

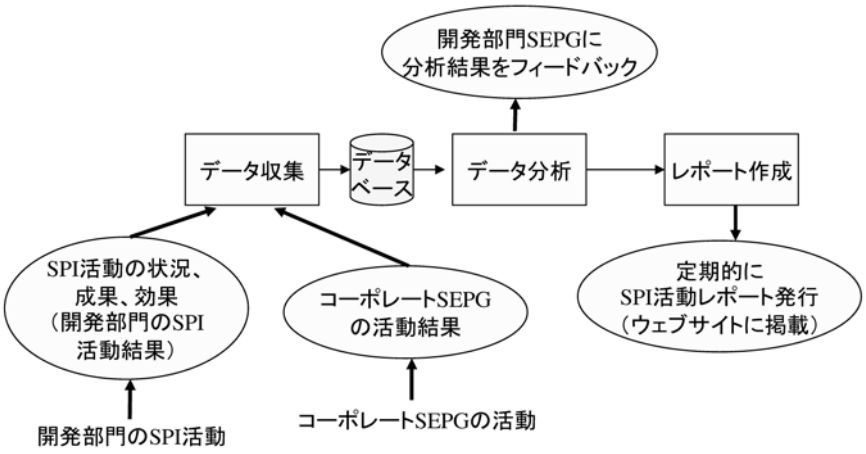


図 3.14: SPI 活動レポート発行までの流れ

明している．また，SPI 活動の効果として，開発期間，開発工数，開発規模の予定と実績を毎期積み上げ，バラツキの傾向の推移を提示している．例えば，開発期間の予定と実績のバラツキが小さくなることは，企業全体として，見積り精度が向上し，予定した納期にリリースができるようになってきたということを示している．

3.7 構築した東芝版 SPI フレームワーク

全社的に SPI 活動を推進・定着させるための仕組みとして提案した SPI フレームワーク（図 3.4 参照）をベースに，前節で示した SPI フレームワークを構成する各要素を実装し，東芝グループ版の SPI フレームワークとして構築した．

東芝版 SPI フレームワークでは，図 3.4 で示した SPI フレームワークを構成する各構成

要素を，表 3.4 に示した用語に置き換えて表現している．

表 3.4: 東芝版 SPI フレームワークにおける構成要素の表現方法

番号	SPI フレームワークの構成要素	東芝版 SPI フレームワークでの表現
活動 1	改善活動を実施する能力	「改善のモデル」の選択と習得
活動 2	推進体制を構築して改善活動を実践	「プロセス改善活動の推進体制」の構築
活動 3	改善活動の進め方に関する能力	「改善の技術とスキル」の習得
活動 4	管理手法／管理ツールを調査して選定する能力	「管理手法／管理ツール」の展開
活動 5	改善事例の提供と他部門の改善事例の利用	「プロセス改善活動推進のための情報基盤」の確立
活動 6	改善活動の成果を論理的に説明	「改善活動の成果と効果」の可視化

構築した東芝版 SPI フレームワークを，図 3.15 に示す．

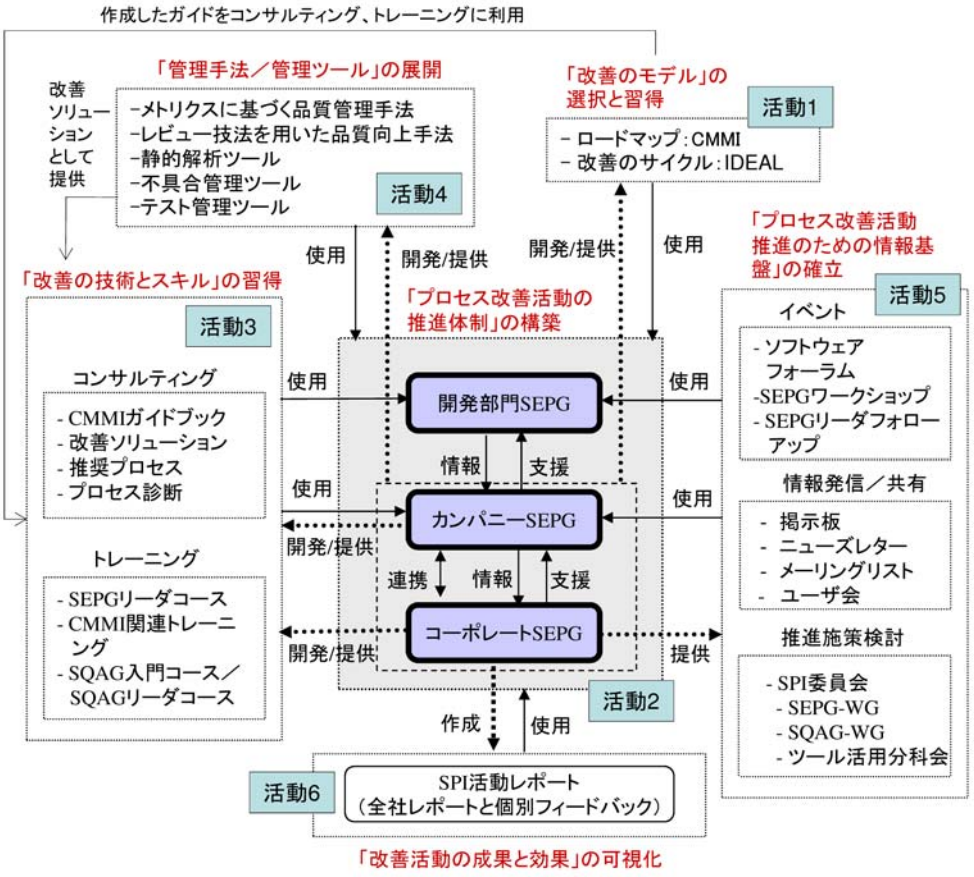


図 3.15: SPI 活動推進のために構築した SPI フレームワーク

東芝版 SPI フレームワークの中核は，活動 2 で構築した 3 階層 SEPG による推進体制である．開発部門における SPI 活動を，カンパニー SEPG とコーポレート SEPG が支援す

る．開発部門の改善に対する能力が十分でない場合や，リソースが十分に確保できない場合でも，カンパニー SEPG とコーポレート SEPG の支援を受けることで，継続して SPI 活動が行える．

開発部門における SPI 活動は，図 2.5 に示した「課題解決型」で実施することもあれば，「改善モデル型」で実施することもある．どちらの場合でも，活動 1 で作成した CMM と IDEAL に関するガイドを有効に活用できる．

活動 1 で作成したガイドは，活動 3 で提供するコンサルティングやトレーニングで活用する．これらのガイドは，モデルの改訂に追従するとともに，東芝グループ内での SPI 活動のノウハウや実績が反映されるように，カンパニー SEPG とコーポレート SEPG によって定期的に改訂されている．

特定の問題領域に対して改善をする場合には，活動 4 で整備した改善ソリューションが活用できる．個々の改善ソリューションでは，ツールや技法以外にも，標準的な導入プロセスやトレーニングコースを整備している．

SPI 活動の成果や結果は，活動 5 で提供している情報基盤を活用して東芝グループ内に展開される．さらに，イベントや委員会での発表資料や成果物は，データベース化され，東芝グループの従業員であれば誰でもアクセスできるようになっている．コーポレート SEPG が計画する全社施策に対しては，カンパニー SEPG のメンバから構成される SPI 委員会がレビュー機能を持ち，妥当性やスケジュールなどの観点からチェックされる．

各開発部門における SPI 活動に関連したデータは定期的に収集・報告され，全社版の SPI 活動レポートとして発行される．さらに，コーポレート SEPG は，各開発部門のデータを定期的に受け取り蓄積しているので，これら経年のデータを活用し，開発部門向けに個別フィードバックを行っている．

コーポレート SEPG，カンパニー SEPG の支援内容と頻度は[10]，開発部門における SPI 活動の定着度合いや成熟度の達成状況に応じて変化する．個々の開発部門が自立的に SPI 活動を推進できるようになると，活動 3〜5 で提供しているさまざまな取り組みや成果物を有効活用できる．また，活動 6 で提供する情報を利用し，さまざまな階層のメンバに活動状況や実績を適切に示すことができる．

3.8 東芝版 SPI フレームワークの活用方法

プロセス改善活動を推進する主体は開発部門である．開発部門は，東芝版 SPI フレームワークで提供している「活動 1:改善のモデル」と「活動 3:改善の技術とスキル」,「活動 4:管理手法／管理ツール」を利用して改善活動を推進する．その改善活動は，組織全体でのプロセス改善活動を推進するための推進体制に支えられている．さらに，プロセス改善の

[10] コーポレート SEPG とカンパニー SEPG については，3.6.2 節を参照のこと

ための情報基盤を共有し，世の中の情報や，他のソフトウェア開発部門での実施状況，ベストプラクティスなどを得る．さらに，改善活動の成果と効果を示し，その結果をスポンサーが確認し，組織全体でのプロセス改善活動に継続的にコミットメントを与える，というステップとなる．

この基本的なステップを開発部門で回すために，東芝版 SPI フレームワークでは，改善のサイクルとして IDEAL を採用している．開発部門における SPI 活動はこの IDEAL のサイクルを回すことを基本としている．

IDEAL のサイクルを基本とするが，改善活動のタイプによっては，東芝版 SPI フレームで提供している手法や活動の使い方などが違ってくる．

本節では，東芝版 SPI フレームワークの基本的な活用方法，アプローチ別の改善活動の進め方，改善活動に対するモード別の進め方，改善活動に初めて着手する場合の進め方について説明する．

3.8.1 基本的な活用方法

各開発部門における SPI 活動は，図 3.16 に示すサイクルで回すことを基本としている．

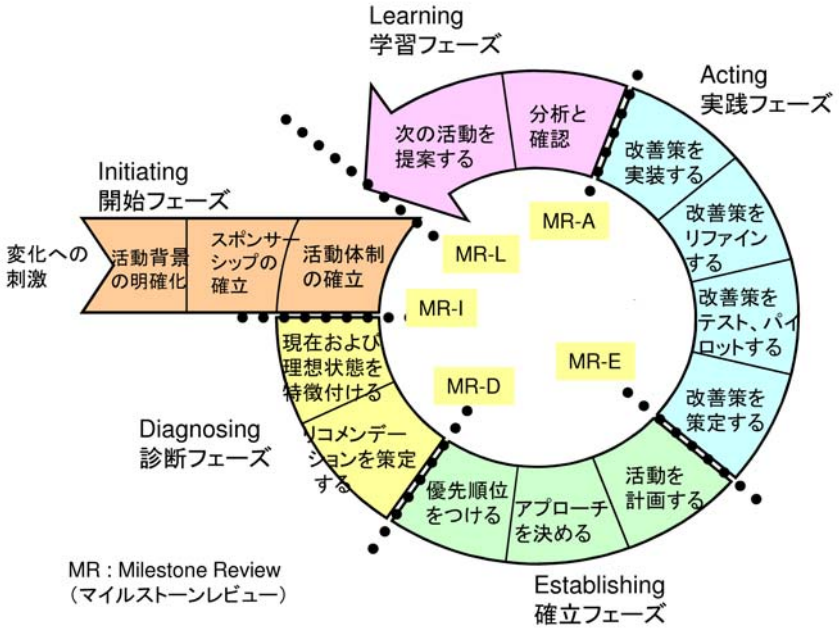


図 3.16: SPI 活動を実施するための基本のサイクル

図中の MR はマイルストーンレビューのことであり，各フェーズの節目で，そのフェーズの活動に対して評価を行い，必要に応じて計画を変更するなどの対策を立てる．

SPI 活動を進めていくうえでは，戦略レベルと戦術レベルの二つのレベルを意識するこ

とを重視している．開始フェーズと学習フェーズは，戦略レベルのフェーズであり，診断フェーズ，確立フェーズ，実践フェーズは戦術レベルのフェーズである．戦略レベルでは，改善活動を大局的に捉え活動をコントロールする．戦術レベルでは，個々の改善を実践する．すなわち，戦略レベルでは，スポンサーや管理層との調整が多く，戦術レベルでは開発現場との調整が多くなってくる．

この IDEAL のサイクルをより効果的に活用するために，マイルストーンレビューで開始／完了基準の観点を明示している．表 3.5 に，各フェーズでの確認項目を示す．

表 3.5: IDEAL の各フェーズにおける確認の観点

フェーズ	開始基準	終了基準
開始フェーズ (Initiating)	・重要なビジネスニーズとしてプロセス改善が必要 ・変化に対する何らかの刺激	・SEPG の組織化 ・スポンサーシップの獲得 ・SEPG 教育 ・SPI の動機が組織のビジネス目標とリンク ・SPI の大目標の設定 ・改善活動企画書作成
診断フェーズ (Diagnosing)	成果物 ・診断報告書 ・改善基本方針	定量的完了基準（参考値） ・SEPG 検討総工数（SEPG の人数） ≧ 60 人 H（≧ 7 人），≧ 40 人 H（＜ 7 人） ・診断報告会の参加率 ≧ 80% （スポンサーの参加は必須） ・改善提案項目数≧ 10
確立フェーズ (Establishing)	成果物 ・改善実践計画書 （中日程計画を含む）	定量的完了基準（参考値） ・改善実践計画書の範囲≦ 8ヶ月 ・改善項目数≧ 3
実践フェーズ (Acting)	成果物 ・改善実施報告書	定量的完了基準（参考値） ・実施できた改善提案の比率≧ 60% ・新プロセスの遵守率≧ 75% ・パイロット実施プロジェクト数≧ 5
学習フェーズ (Learning)	成果物 ・活動報告書	定量的完了基準（参考値） ・得られた教訓の数≧ 10

3.8.2 改善活動のアプローチ別の進め方

図 3.17 に，改善活動のアプローチ別の改善活動に対するスタンスを分類したものを示す．これは，参考文献 [43, 乗松] を参考にして整理したものである．これを SPI 活動の計画立案時に活用し，自分たちがどの領域にあるのかを認識してもらうようにしている．そのことで，活動の進め方や改善施策の検討方法といった計画立案にフィードバックがかかる．

図 3.17 は，横軸に改善活動のタイプとしてモデル重視なのか問題解決重視なのかを配置している．縦軸には，組織としての改善活動の経験を配置している．この両軸の中で，改善のスタンスを，表 3.6 に示す 6 つのタイプに分類している．

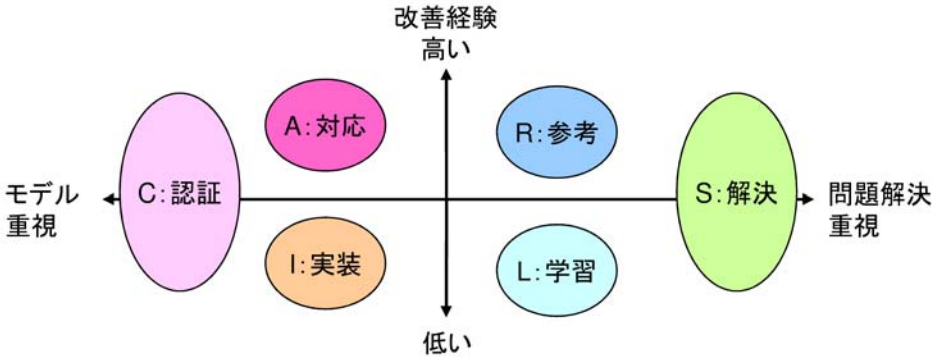


図 3.17: アプローチ別の改善のスタンス

表 3.6: IDEAL の各フェーズにおける確認の観点

分類	アプローチ
認証型 (Certification)	・モデルによる「認証」「認定」を得る
対応型 (Adaptation)	・現在の活動・資産をモデルの期待に「対応」させる ・モデルによるアプレイザルを行い，ギャップに対応するように，現状を修正する
実装型 (Implementation)	・モデルに合わせて組織標準として「実装」する ・モデルを解釈したうえで手順を構築する
参考型 (Reference)	・改善活動の一環として，モデルを「参考」にする ・組織標準とモデルとのギャップを分析して，組織の弱みを元に改善提案を行う
学習型 (Learning)	・モデルを通じて，ベストプラクティスを「学習」する ・役に立つ所を選択しながら，取り入れる
解決型 (Solution)	・モデルを参考にせずに問題解決を行う

3.8.3 改善活動に対するモード別の進め方

SPI 活動を進める際，前節で示した分類だけではなく，組織メンバの改善活動に対するモード（雰囲気）に目を向けることが重要である．例えば，過去にトップダウンで改善活動を強要され，効果が確認できなかったような場合には，改善活動自体に懐疑的な立場を取ることが多い．このような場合には，組織メンバの意識を変える必要がある．

以下の表 3.7 は，組織の改善活動に対するモード（雰囲気）を懐疑的，消極的，協調的，積極的の 4 つに分類し，それぞれのモードにおける「組織の状態」,「モード遷移時のトリガ」,「支援の施策（例）」を整理したものである [54].

3.8.4 改善活動に初めて着手する場合の進め方

組織長のリーダシップのもと SPI 活動をすることが決まり，SEPG リーダが任命されるという状況はよく見受けられる．この時，SEPG リーダに任命された方は，改善活動の経

表 3.7: 組織の改善活動に対するモード別のアプローチ

組織モード	懐疑	消極	協調	積極
組織の状態	改善活動自体が余計な取り組み、どうせやらないオーラ	改善は致し方ない プロセスは邪魔、障壁	プロセスは大事という意識 自作は躊躇	自分たちでプロセスを決めて、守る
モード遷移のトリガー	場作り コミュニケーション チームビルド	身近な課題の解決 信頼感醸成	プロセス志向の改善	（維持継続） 自主的、効果的な取り組み（オーナーシップ確立）
支援の施策（例）	SEPG会議の開催 1点突破型小規模改善（ツール活用など）	ワーキンググループなどへの介入による手厚い支援活動 代表プロセス作り	プロセス全体像（土台）の構築 プロセスに関する議論 権限委譲	プロジェクト課題解決から組織的なプロセス改善へ SQA活動の充実

験がなければ，何をすれば良いかわからないという状態になる．このように，初めて組織で改善活動を開始する場合の進め方の推奨例を図 3.18 に示す．

1サイクル目

フェーズ	活動内容
開始	SEPGリーダを任命する コンサルを入れて診断することを了解する
診断	コンサル中心に広くヒアリングを実施する 診断結果を共有し，現状認識を合わせる
確立	課題の優先順位付けと今後の戦略を検討する
実践	スキップ
学習	これまでの活動の振り返り

2サイクル目

フェーズ		活動内容
開始		SEPGメンバをアサインする 優先度の高い課題ごとにTWG（Technical Working Group）を割り当てる
ワーキンググループごとに実施	診断	1サイクル目のヒアリング結果を見直し，課題を再確認する
	確立	改善アプローチを複数考え，リソース，効果などから有効なアプローチを選択する
	実践	改善策を立案し，パイロッティングする
学習		これまでの活動の振り返り

3サイクル目

フェーズ		活動内容
開始		新しいワーキンググループがあればリソースを割り当てる
ワーキンググループごとに実施	診断	パイロットで把握した課題を確認する 制度化に必要な課題を検討する
	確立	解決できる課題と，妥協しなくてはならない課題を分類する
	実践	解決できる課題を改善し，本番適用する
学習		これまでの活動の振り返り

図 3.18: 改善活動に初めて着手する場合の進め方

図 3.19 は，図 3.18 に示した 1 サイクル～3 サイクルまでを整理した結果である．IDEAL ではモデル化するため単純なサイクルになっているが，実際の SPI 活動では，課題が階層化している事があり，IDEAL のサイクルも入れ子の状態になることも多い．

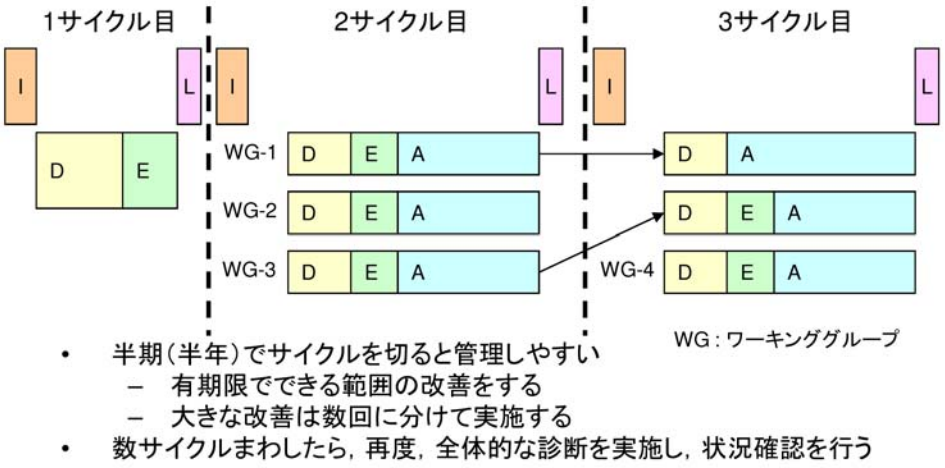


図 3.19: 改善のサイクルの回し方

第4章　全社的なSPI活動の実践結果

本章では，1.2 節で示した目的「企業内において組織横断的に SPI 活動を推進・展開するための方法論」の確立を目的として提案した，SPI 活動を推進・定着されるための仕組みである「SPI フレームワーク」に基づいて，東芝版 SPI フレームワークを構築し，そのフレームワークを維持・発展させながら約 10 年間の長期にわたり実践した結果を示す.

4.1　適用対象組織の概要

2000 年から開始した全社的な SPI 活動は，筆者が所属する会社およびグループ会社を含めた範囲を対象としている．この対象範囲の中で，SPI 活動の導入・定着を目指した組織は約 70 部門であった[1]．対象とした組織の概要を表 4.1 に示す.

表 4.1: 対象とした組織の概要

組織の分類	コーポレート	グループ会社
デジタルプロダクツ	8	5
電子デバイス	2	1
社会インフラ	21	10
ソリューション	-	8
海外開発拠点	-	4
研究・開発	6	1
合計	37	34

また，表 4.1 で示した開発部門に所属する人数の分布は，図 4.1 に示すとおりである[2].

SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」に基づいて東芝版 SPI フレームワークを構築し，そのフレームワークを活用して約 10 年間の長期にわたり実践した結果は，表 4.1 に示したとおり，さまざまな製品領域を持つ開発部門を対象としたものであることが分かる．さらに，開発部門の組織人員数も，図 4.1 に示すとおり，1〜50 人の小規模組織から 201 人以上の大規模組織までをカバーしている．筆者が所属する企業ではカンパニー制を取っており，それぞれのカンパニーが独自の開発プロセスと組

[1]本活動開始時点から本論文作成時点までは約 10 年間が経過しているが，開発部門数としてはそれほど大きな差は出ていない.

[2]このデータは，全体の約 1/3 からのデータに基づいて作成しているが，全データを使った場合でも，おおよそ同じ割合になることを確認している.

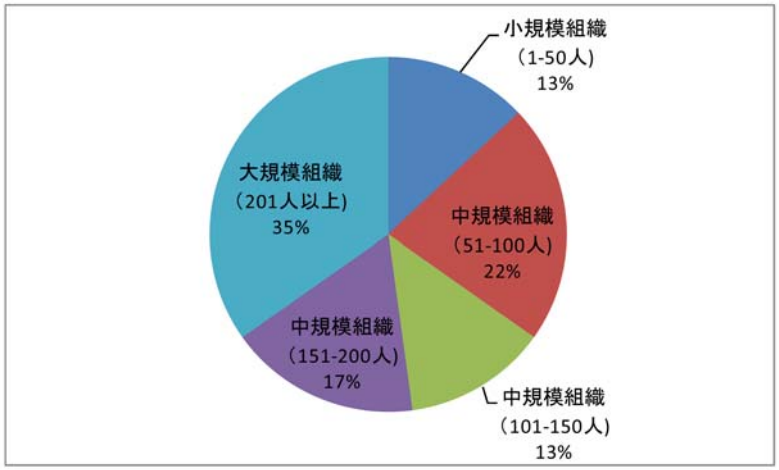


図 4.1: 対象組織に所属する人数の分布

織文化を持っている．このことは，本研究による実践結果が，筆者の所属する企業独特のものであるということではなく，ある程度の一般性を与えるものであることを示している．

4.2 全社的な SPI 活動の推進方針

3.1 節で述べたように，この活動の開始時は，ソフトウェア／ハードウェアの開発量が大幅に増大したことで，以下に示すことの実現が強く求められていた．

- 開発量の増大に対応できる設計効率の実現
- 設計上流での設計品質の向上
- 設計期間の短縮と適切な市場投入のタイミング

これらを実現するための一つのアプローチとして組織力を強化して対応するという方針が示され，以下を実現することが目的として掲げられた．

1. ソフト生産技術力の向上
2. 改善のできる組織文化の構築

目的 (1) に対しては，組織力を強化することで，東芝グループ全体としての組織の能力および体質が改善され，結果として組織全体の生産性ならびに品質の向上に寄与することで達成できると考えた．この目的の達成度を定量的に評価する方法としては，3.6.6 項で示した SPI 活動レポート発行のための活動で蓄積されたデータを活用し，この活動の「成果」と「効果」を示すことにした．「成果」と「効果」の基本的な考え方を以下に示す．

- SPI 活動の成果は，SPI 活動の定着状況や組織成熟度の達成状況で確認する

- SPI 活動の効果は，プロジェクトの QCD（Q : 不具合数，後戻り工数，C : 開発工数，開発規模，D : 開発期間）の予定と実績を毎期積み上げ，その傾向の推移で確認する
  - 予定と実績のバラツキが縮小することは，企業全体として見た場合，見積り精度が向上し，予定した納期にリリースができるようになってきたということを示している．

上記の考え方は，全社レベルでは品質指標（例えば，開発規模当たりの不具合件数といった）では効果を評価しないということを示している．一方，個々の開発部門に対しては，予実差のバラツキ以外にも品質指標の推移を確認することで，QCD が向上しているのか悪化しているのか確認することを推奨した．

目的 (2) に対しては，東芝グループ内のソフトウェア開発部門の SPI 活動実施率を指標として捉えることにした．活動開始当初，東芝グループ内で SPI 活動が必要と見積もった開発部門数は約 70 部門であった．このうち，少なくとも 80% の部門で SPI 活動が定着していることを目指した．定着の定義は難しいが，ここでは，第三者（コーポレート SEPG，カンパニー SEPG）から見て開発部門の中で定常的に SPI 活動が実施されている状態であれば「定着」と判断することにした．

90 年代に全社的なソフトウェア開発の生産性向上に関する活動が実施され，事務局主導で進められた[3]．その結果，東芝グループ全体へは網羅的に施策の展開はできたが，時間の経過とともにやらされ感が出てきて，本質的な活動につながらなかったことがあった．この経験を踏まえ，本活動を開始する際には，以下の方針を示した．

- 東芝版 SPI フレームワークは永続的に維持・発展させる
- このフレームワークを使って SPI 活動を推進したい開発部門には，コーポレート SEPG とカンパニー SEPG が積極的に支援する
- 開発部門長のコミットメントを重視する
  - 開発部門長の SPI 活動に対するコミットメントの提示を必須とする

このような方針を示したうえで，東芝版 SPI フレームワークを活用して東芝グループ内にこの活動の存在感を示し，SPI 活動に着手する部門を増やすことにした．事務局主体の全社運動であればすべての開発部門に対して指示を出すことができるが，この方針では，すべての開発部門まで浸透させることは難しい．今回の活動では，すべての開発部門へ展開するのではなく，開発部門長のコミットメントを得たやる気のある開発部門へ展開することを優先した．

[3] ここでは，施策を展開し，その活動結果を報告してもらうことが中心であり，活動の直接的支援が薄いことを事務局主導と呼んでいる

4.3 全社的な SPI 活動の推進方法

3.1 節，4.2 節で示したとおり，本活動の目的としてコーポレートの上級管理層からは，「改善のできる組織文化の構築」が掲げられた．一般に，成熟度レベルを 1 つ上げるためには，おおよそ 2 年前後の期間が必要と言われている [34]．つまり，レベル 1 からスタートした場合，レベル 5 まで到達するには，少なくとも 8 年間かかるということである．全社のソフトウェア開発部門に対して長期に支援を継続するためには，推進側の組織でも，推進担当者のリソース確保や能力向上などの中長期の計画が必要となる．そこで，全社に向けて SPI 活動を展開する際，計画，実装，展開，定着の 4 つのフェーズに分けた約 10 年間にわたる中長期計画を立案した（表 4.2）．

表 4.2: SPI 活動を推進する部門（コーポレート SEPG）の推進計画

		推進フェーズと年度			
		計画	実施（基礎固め）	展開	定着
		2000–2002	2003–2004	2005–2007	2008–2010
全社的にSPI活動を推進する部門の推進計画	開発部門に対する支援方法	数部門でアプレイザルを実施し，CMMIをベースにしたSPI活動を実施（先行評価）	主要な開発部門に対する手厚い支援によるSPI活動の着実な推進	・全社の開発部門全体をカバーする範囲までSPI活動を展開 ・SPI活動が定常的に行える部門に対しては，自立化を促進	・SPI活動は各開発部門で自立的に実施．間接的な方法で開発部門のSPI活動を支援 ・次の中長期計画に向けた計画の立案
	支援メンバのスキル強化	CMMIとSPI活動推進に関する基本スキルの習得	専門領域の明確化と専門家の育成	専門性の深掘りと経験やノウハウの形式知化	開発部門へ深く入り込んで現場経験を蓄積
	支援メンバのリソース計画	コアメンバの選定	支援部門の増加にともなうリソースの拡大と支援担当者の育成	・リソースを維持してSPI活動を展開 ・開発部門をグループ化し，グループ毎にチームを作って支援	開発部門のSPI活動の自立化にともないリソースを縮小

計画，実施フェーズでは，SPI 活動を全社的に支援する部門側のリソースを確保するとともに，支援担当として保持しておくべき技術を習得する．さらに，全社レベルで SPI 活動を推進するための体制を構築する．また，この活動を推進するために必要となる方法論やツールを開発し，基本的なトレーニングコースなどを整備する．開発部門における SPI 活動の支援は，CMMI の段階表現を利用した診断結果をベースに，改善項目の選択と優先度付けを行い，改善活動を進めることを基本とする．改善活動の進め方にはさまざまなバリエーションがあるが，まずは基礎を固めるという意味で，モデルベースの SPI 活動を主体に実施することにした．

展開フェーズでは，支援する開発部門が増えてくるので，モデルベースの SPI 活動だけ

ではなく，開発部門の特徴に合わせた改善方法を提供することが必要となる．さまざまなバリエーションの改善方法を提供するために（例えば，CMMI の連続表現を用いた改善や課題解決型の改善など），全社の SPI 活動を推進する部門では，SPI 活動推進に関する技術開発やノウハウの形式知化は必須である．開発部門における SPI 活動を支援する際には，改善項目の内容やスケジュールを検討するために，開発部門で SPI 活動を推進しているメンバと推進部門側のメンバとで，定期的な打合せを行うことを基本とする．推進部門側の担当者は，この期間は，支援部門以外での診断活動や強化すべき 6 つの活動のいずれかを担うことがある．支援の継続性を確実にするため，開発部門毎にグループ化し，そのグループに対して複数名でチームを作り，情報交換を密にして支援するという体制を構築した．

定着のフェーズでは，開発部門が SPI 活動を自立的に推進できるようになってくる．したがって，段階的に，開発部門における SPI 活動を支援するためのリソースを減らし，それ以外の活動にリソースを割り振るとともに，定着後の次の中長期計画の策定に着手する．

4.4 東芝版 SPI フレームワークを構成する各要素の実績

東芝版 SPI フレームワークをベースに，全社的な SPI 活動を 10 年以上にわたり実践してきた．本節では，このフレームワークを構成する各要素の実績を示す．

4.4.1 活動 1 :「改善のモデル」の選択と習得に対する実績

SEPG リーダコースは毎年開催され，15〜20 名前後の受講生が参加する（参加者数の累積は図 4.4 を参照のこと）．このトレーニングコースを開催するにあたり，毎年教材を改訂している．CMM/CMMI ガイドや IDEAL ガイドも，コーポレート SEPG とカンパニー SEPG の SPI 活動の実践経験をフィードバックして改訂される．

東芝グループ内での SPI 活動が浸透してきた段階で，この活動をしっかりと根付かせるために，SPI 活動推進のための方針と指針を全社規定として定義してもらいたいという意見が，多くのカンパニー，開発部門から寄せられた．カンパニー，開発部門からすれば，SPI フレームワークが維持・改善されていることは分かるが，全社としての確たる方針のもとで進められているわけではない，という不安感があったものと思われる．そこで，SPI と SQA に関する全社規定を制定した．以下に，それぞれの規定の目的，適用範囲を示す．

- ソフトウェア開発ガイドライン 〜ソフトウェアプロセス改善 (SPI) 推進ガイド
  - 1. 目的
    本ガイドラインは，ソフトウェアの設計／開発／保守部門に対して SPI 活動を進めるための指針を示し，各部門での SPI 活動を常態化させてゆくことを目的とする．
    なお，本ガイドラインはソフトウェア部門でのプロセス改善活動の指針を示すものであるが，システム，ハードウェア部門においても，それぞれの組織的な開発能力を向上させるために参考にすると良い．

  - 2. 適用範囲と対象者
    本ガイドラインは，東芝グループにおける SPI 推進活動に適用し，活動に携わる次の人を対象者とする．
    * 統括技師長／技師長，またはそれに代わる技術トップ
    * ソフトウェアの設計／開発／保守を行う部門とその構成員
    * 技術管理部門
- 東芝グループ ソフトウェア品質保証（ガイドライン）
  - 1. 目的
    本ガイドラインは，ソフトウェア設計開発部門での上流工程からの品質作り込みを促進し，品質・コスト・納期 (QCD) を源流から改善するための仕組みの構築を示したものである．
  - 2. 適用範囲
    本ガイドラインの適用範囲は，株式会社東芝ならびに株式会社東芝が経営権を留保していると認められる関係会社 (以下東芝グループという) とする．

上述したガイドライン（規定）は CMM や IDEAL といったモデルを説明したものではないが，内容には，これらの改善モデルを有効に活用することという方針が述べられている．全社的な SPI 活動を推進する際に，CMM と IDEAL という改善モデルを選択し，それらのモデルの利用が全社レベルの規定として認められた．

4.4.2 活動 2 :「SPI 活動の推進体制」の構築に対する実績

図 4.2 は，SEPG 設立部門数の推移を示したものである．図 4.2 に示したように，2000 年度に活動を開始後，2008 年時点で，開発部門 SEPG の数は約 60 部門となっている．また，カンパニー SEPG もほぼ組織化が完了し，3 階層 SEPG による体制が確立された．この結果から，全社レベルでの SPI 活動推進のための体制が構築され，維持されていることが分かる．

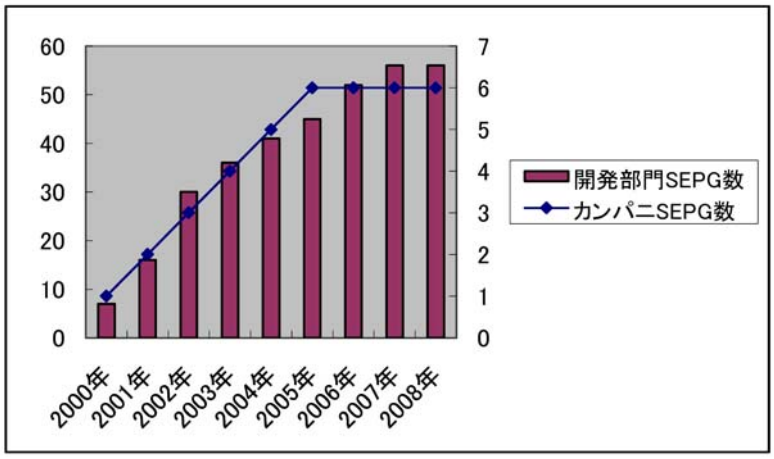


図 4.2: SEPG 設立部門数の推移

全社的な推進体制を図 4.3 に示す．全社的な SPI 活動の推進をより強固なものとするために，SPI 委員会を設置し，その委員会の下に SEPG 分科会，SQAG 分科会，ツール活

用分科会を設置した．各分科会では，WG(Working Group) などを設置し，具体的な実践事例の交換や各カンパニーが抱えている課題の解決を図るとともに，東芝版 SPI フレームワークで活用するトレーニングコースや改善ソリューションとして提供する技術開発などを実施する．親委員会である SPI 委員会では，各カンパニーの活動状況や施策内容の確認を行うとともに，コーポレート SEPG が立案する施策に対するレビューを行う．このような体制が確立できたことで，全社レベルで SPI 活動を推進するというコミットメントがより強化されてきた．

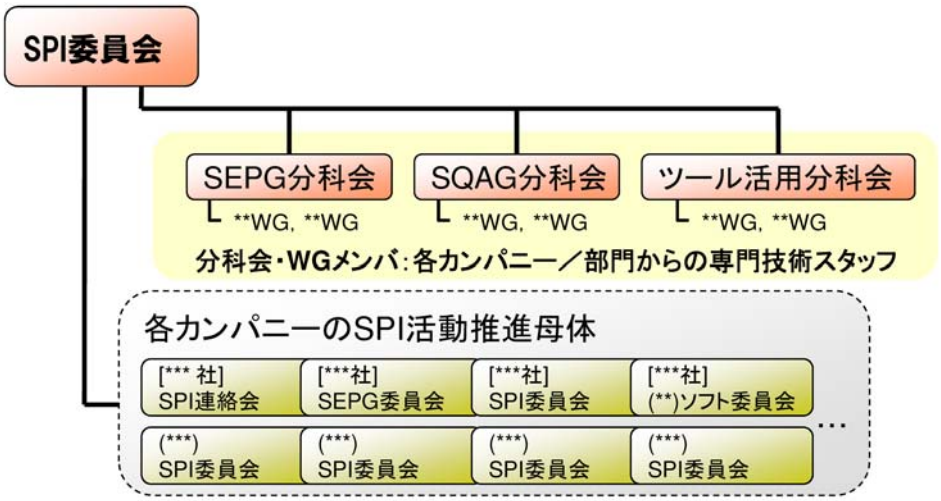


図 4.3: SPI 委員会の設置

また，各カンパニーでも，カンパニーレベルで SPI 活動を推進する母体として委員会や連絡会を設置するようになってきた．

4.4.3 活動 3 :「改善の技術とスキル」の習得に対する実績

CMM 関連のトレーニングを活動開始当初から提供し，その後，SEPG リーダコース，SQAG 関連コースを提供してきた．図 4.4 に，CMMI 関連教育，SEPG リーダコース，SQAG 関連教育の受講者数の累計を示す．CMMI 関連教育の累計の受講者数は 1,000 人を超えている．また，SEPG リーダコースには，毎年，約 20 名の受講者が参加しており，累計の受講者数は 150 名を超える．SEPG リーダコースには，同一部門からの受講者も多く，SEPG リーダを担うことができる人材の育成が計画的に進められている．

開発部門内で SPI 活動を推進する際には，SEPG は改善活動を推進し，SQAG はプロジェクトや組織の開発プロセスのチェックや開発成果物のレビューなどを担うという体制を構築している．開発部門内で SPI 活動が継続的に行われ，開発プロセスが整備されてくると，SQAG に割り当てられる担当者も多くなり，SQA に関連したトレーニングの整備

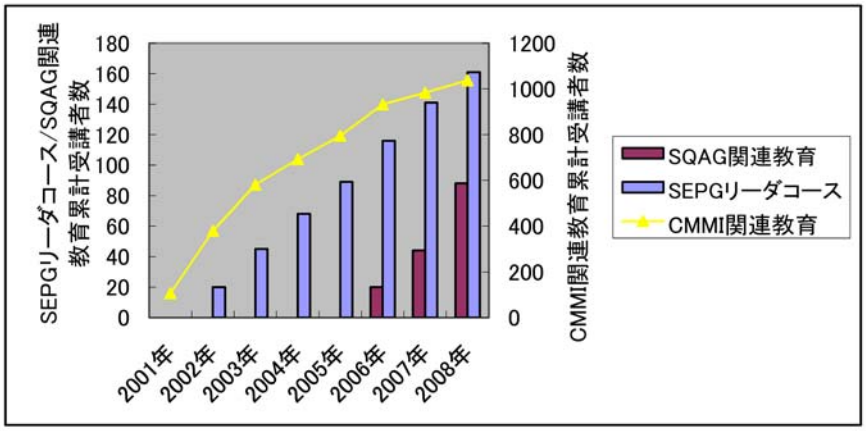


図 4.4: トレーニングコースの累計受講者数

が求められてきた．そこで，2006 年から SQAG 入門コース，2008 年から SQAG リーダコースを提供し，SQA 活動を遂行できる人材の育成も進めている．

4.4.4 活動 4 :「管理手法／管理ツール」の展開に対する実績

3.6.4 項で，改善ソリューションとして提供している管理手法／管理ツールを示した．また，これらの手法やツールを実際に提供するまでのステップを図 3.12 に示した．

ここでは，図 3.12 に示したステップ 2 とステップ 3 の取り組み事例を実績の一つとして説明する．ステップ 2 とステップ 3 の概要，およびそれぞれのステップの事例として説明する項目を以下に示す[4]．

- ステップ 2：運用プロセスの構築
  - 概要：開発した手法／ツールを効果的・効率的に活用するための運用プロセスを定義する
  - 事例：メトリクス計測結果のフィードバック方法とシステム化
- ステップ 3：体制と導入手順の整備
  - 概要：体制と導入手順を整備して，より多くの開発部門に提供する
  - 事例：メトリクス計測／プログラム静的解析システムの導入手順

メトリクス計測結果のフィードバック方法とシステム化

付録 A と付録 B では，図 3.12 に示した手法やツールを実際に提供する際のステップ 1「手法／ツールの分析や開発」の内容を示している．どちらの結果からも，手法やツールの

[4]ステップ 1 は「手法／ツールの分析や開発」であり，手法／ツールを活用するために必要なデータ分析方法の検討や機能開発を実施する．このステップ 1 の事例は，付録 A，付録 B を参照のこと

有効性をデータ分析などにより確認し，効果的な活用方法を提示するには，それ相応の労力が必要となることが理解できる．開発部門でこの作業を担うのはリソース的に難しい場合が多いため，このような活動は，カンパニーやコーポレートで対応することが望ましい．

SPI 活動において重要な役割を担う管理手法や管理ツールの導入・定着に向けて，次のステップとして必要なことは，効果的・効率的に活用するための運用プロセスを定義することと，自動化を促進するためにシステム化することである．

付録 A「メトリクスを活用した品質管理手法」，付録 B「プログラム静的解析技術の効果的な活用方法」で示したとおり，メトリクス計測とプログラム静的解析は定期的に実行し，その結果をレビューして開発やテストの作業にフィードバックすることの有効性が確認できている．また，品質基準値や重大な警告ランクを利用してレビュー対象を特定することが，レビューの効率と効果を高めることが分かった．このような結果から，開発部門での基本的な運用プロセスを図 4.5 のとおり定義した．

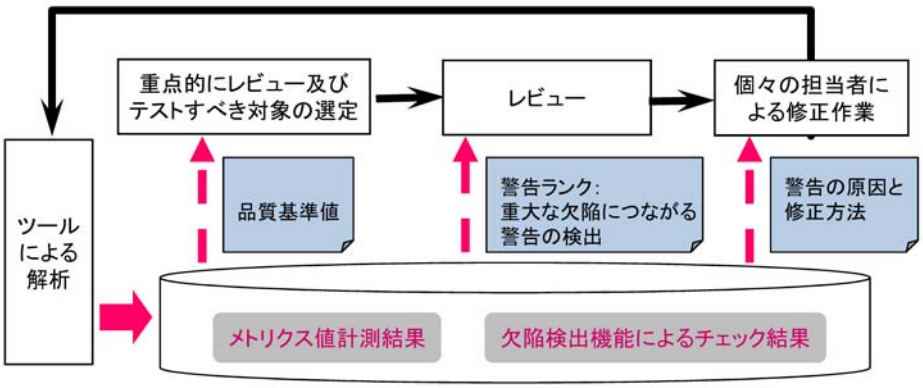


図 4.5: メトリクス計測／静的解析の基本的な運用プロセス

図 4.5 に示す運用フローが決まっても，品質基準値を超えたものだけを抽出する，設定した警告ランクのものをだけを抽出する，警告の原因と修正方法をタイムリーに提示するということができなければ，実際の開発で活用することは難しい．そこで，解析結果から必要な情報を抽出するための条件の設定，条件に合ったものだけを抽出，修正方法の提示といったことをサポートするシステムを開発した [85]．開発システムの全体像を図 4.6 に示す．

このシステムでは，さまざまなプログラミング言語の解析ができるように，解析エンジンとして付録 A，付録 B で説明したツール以外のものも組み込めるようになっている．このシステムの実行プロセスを以下に示す．

1. ソースコードの取得

   構成管理下に登録されているソースコードを，利用者が設定したタイミングで自動

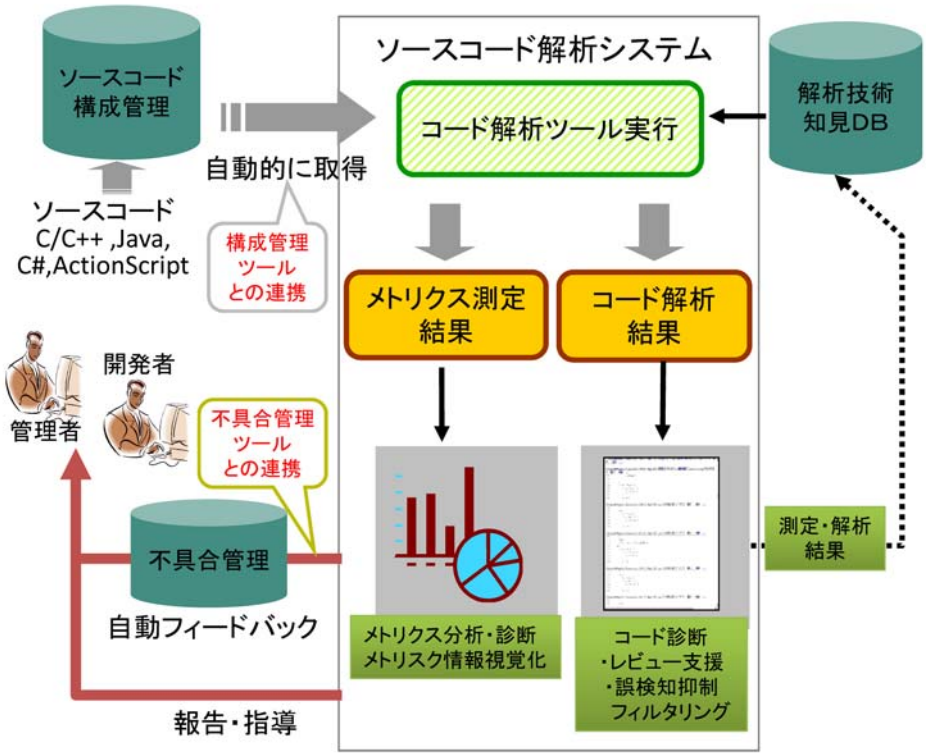


図 4.6: メトリクス計測／静的解析システム

的に取得し，計測と解析を行う．

2. 計測と解析結果のまとめ

   計測されたメトリクスと静的解析の結果は，それぞれ，メトリクス測定結果レポート，コード解析結果レポートとして提示される．この結果は，Web を介して参照できる．ここでは，過去の測定結果や解析結果から得られている知見を活用して，レビュー対象とすべき計測結果や解析結果が抽出される．

3. 不具合管理ツールへの登録

   計測された結果あるいは静的解析結果から，危険な部分が検出された場合には，不具合管理ツールと連動し，開発担当者や管理者に，不具合の可能性が高いという情報を発信する．

4. レビュー結果のフィードバック

   不具合管理ツールに登録された情報に対する対応結果を登録すると，その内容が解析技術知見 DB に蓄積される．この蓄積された結果は，次のメトリクス計測・静的解析に利用される．

このように解析実行の自動化，設定作業の簡素化及び Web 上での閲覧を可能とすることで効率よくメトリクス計測ツールや積的解析ツールが利用できるようになる．

メトリクス計測／プログラム静的解析システムの導入手順

図 3.12 に示した手法やツールを実際に提供する際のステップ 2 である「運用プロセの構築」の活動事例を前節 4.4.4 で示した．ここまで整備できると導入・展開は容易にできると考えがちであるが，実際には，開発部門の多種多様なプロセスや要望への対応，導入希望の開発部門が同時期に複数重なることへの対応など，まだ検討すべきことが多い．

そこで，手法やツールを実際に提供する際のステップとして「体制と導入手順の整備」を定義している．この活動の目的は，開発部門への管理手法／管理ツールの導入・定着を，所定の期間で確実に達成することと，開発部門の推進担当者とのネットワークを確立し，改善要望の吸い上げ，新規機能の提供などを実施することである．

不具合管理ツールの導入・展開時の経験をふまえ [62]，メトリクスを活用した品質管理手法とプログラム静的解析技術を活用するためのシステムを導入するための手順を，図 4.7 のように定義した．

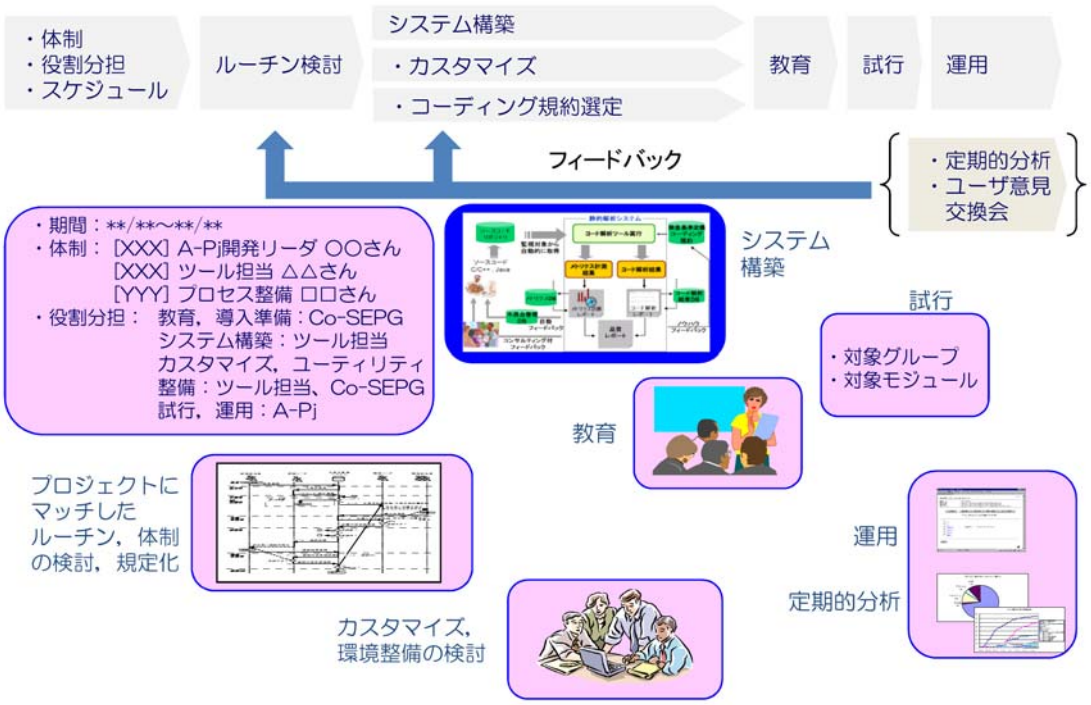


図 4.7: メトリクス計測／静的解析システムの導入手順

システム導入時には，図 4.7 で示した手順にしたがって進めていく．運用開始までおおよそ 1ヶ月の期間でスケジュールを立案することが多い．

プログラミング言語や開発環境の進化にともない，解析エンジンや利用環境も日々進化し続けている．このような状況のもと，高品質なサービスを継続するには，体制面の強化も必要となる．

各開発部門で共通に使える管理手法や管理ツールの普及・展開を図る際には，3 階層 SEPG の構造を活用するとともに，掲示板やユーザ会などを立ち上げて，情報共有と情報発信ができるようにしている．

典型的な推進体制を図 4.8 に示す．

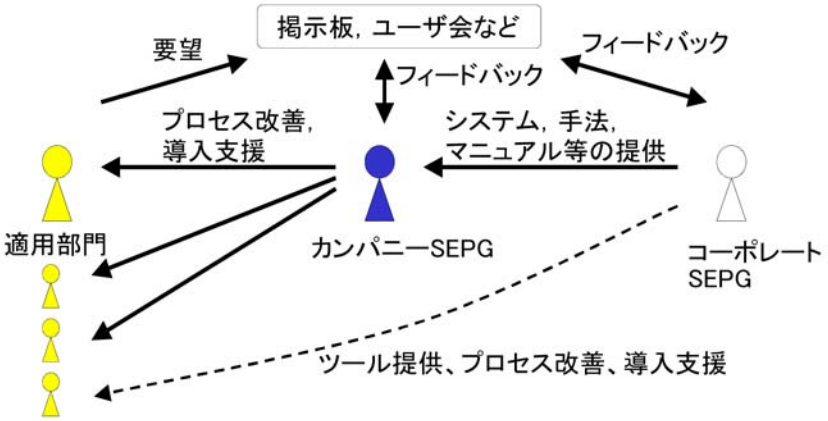


図 4.8: 管理手法／管理ツールを推進するための体制例

図 4.8 に示したとおり，負荷分散のために，カンパニー SEPG と連携した導入支援体制を作り役割分担を行っている．導入依頼がコーポレート SEPG に直接届くこともあるが，できるだけカンパニー SEPG が窓口になってもらえるように推進している．

また，カンパニーや開発部門でシステムの利用環境が構築できない場合には，コーポレート SEPG で準備したシステム環境を利用できるようにしている．このような仕組みを整備しておくことで，自前でシステム環境の構築と運用ができないカンパニーや開発部門でも利用できるようになる．

管理手法／管理ツールの導入実績

改善ソリューションの一つとして展開しているプログラム静的解析システム（メトリクス計測も含む）は，システム化及び導入サポートによる簡便化と導入維持コストの低コスト化により，2007 年度以降，飛躍的な展開を図ることができた．このシステムの導入により，致命的なバグの検出に貢献し，テストでは発見困難であったバグを究明するという実績も上げている．製品サイクルの短いデジタルコンシューマー製品においては，1 千万行を超えるソースコードを週一回解析実施する運用体制を確立した．現在の導入部門数を図 4.9 に示す．

プログラム静的解析システム以外の手法やツールも，図 3.12 で示したステップで改善ソリューションとして仕上げることができた．表 4.3 に，提供している改善ソリューション

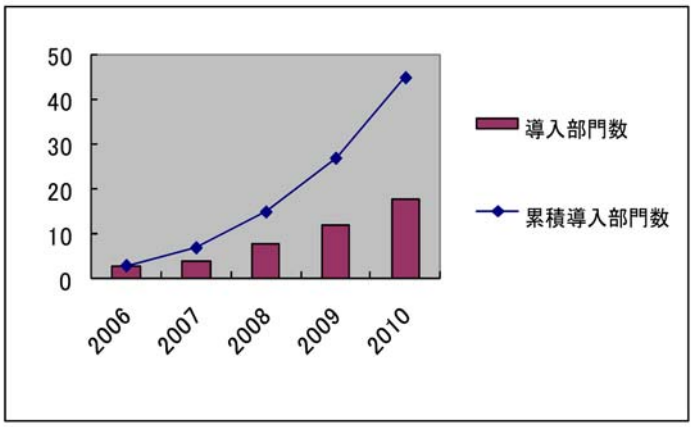


図 4.9: プログラム静的解析システムの導入部門数

の概要と導入実績を示す.

表 4.3: 提供している改善ソリューションの概要と導入実績

改善ソリューション	手法化／ツール化の特徴	運用プロセス上の特徴	体制と導入手順の特徴		導入実績
			体制	導入手順	
レビュー技法を用いた品質向上手法	ピアレビュー，インスペクション技法を取り入れたレビューの方法（レビュー参加メンバの特定，レビューの進め方とレビューの観点を規定）を確立	レビュー結果をデータとして蓄積し，そのデータを活用したプロセスを構築	コーポレートSEPGメンバであれば誰でも対応できるようにトレーニングを実施	開発中の成果物を利用したワークショップを開催し，効果と手法を理解してもらったあと，展開方法を検討	40
不具合管理ツール	・分散開発環境への対応 ・ワークフローを定義できる機能の実装 ・分析機能の強化	標準の不具合管理プロセスをベースにしたカスタマイズ	・ベンダにツールを移管しコンサルティングと販売を依頼 ・コーポレートSEPGは新規機能開発に従事	導入条件検討（体制，役割，スケジュール），ルーチン検討，システム構築，教育，試行，運用のステップで導入	60
プログラム静的解析ツール（メトリクスに基づく品質管理手法を含む）	・品質基準値，重要度の高い警告の設定によるデータ抽出の効率化 ・システムの自動化	メトリクス計測結果と静的解析結果を利用したフィードバックサイクルの提案	コーポレートSEPGに専任メンバを設置		50
テスト管理ツール	・既存の帳票とのエクスポート／インポートの実現 ・テスト実施状況の可視化機能の強化	既存のテストデータを登録し，テスト管理として実現したいことを確認	コーポレートSEPGに専任メンバを設置	試行評価後，必須の機能を確認して作り込みを行い提供	25

4.4.5 活動 5 :「SPI 活動推進のための情報基盤」の確立に対する実績

毎年開催するソフトウェアフォーラムは，社内外の状況を知って刺激を受け，新たな取組みのヒントを発見する場として，SPI 関係者のみならず，ソフトウェア開発者に広く認知されている．2004 年以降は，コンスタントに 200 名以上の参加者があり，全社レベルの

イベントとして認知されている．このフォーラムでは，社外からの講演と社内の事例発表が行われおり，情報共有だけでなく，各開発部門での SPI 活動の活性化に貢献している．

フォーラムにおける事例発表では幅広いテーマが扱われている．文献 [58, Moriya] の「定量的プロジェクト管理の実践」もその一つである．参加者が社内と限定されていることもあり，定量的プロジェクト管理に関する成功事例や失敗事例，ノウハウなどが具体的なレベルまで公開され，社内の定量的プロジェクト管理の展開に大きく寄与している．

図 4.10 に，東芝ソフトウェアフォーラムの参加者数の推移をグラフで示す．このグラフからも分かるとおり，最初の数年で参加者数が増加し，2004 年から 200 名を超える参加者となっている．

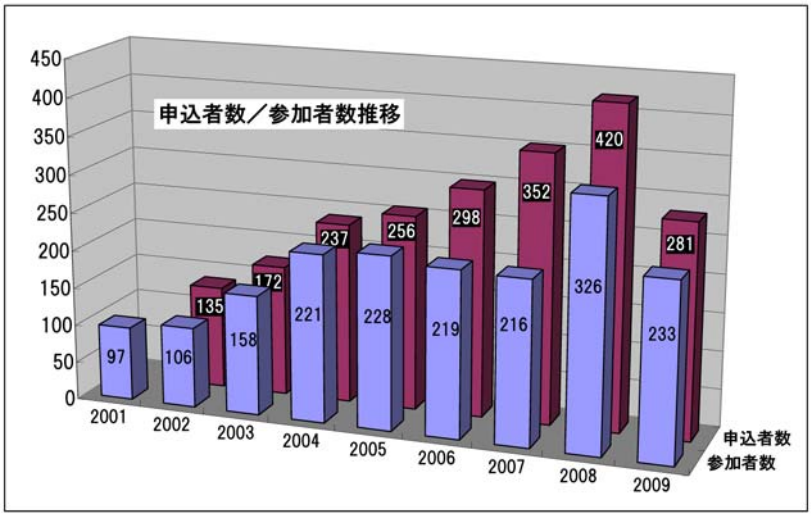


図 4.10: 東芝ソフトウェアフォーラムの参加者数の推移

4.4.6 活動 6 :「改善活動の成果と効果」の可視化に対する実績

図 3.14 で示したデータ収集，分析，レポート作成のステップを 2004 年から開始し，SPI 活動レポートを半期に一回発行している．SPI 活動レポートの目次を図 4.11 に示す．

上記に示したレポートを作成するために，収集したデータを登録するためのデータベースを構築した．データベースにデータを登録すると，全社レポートとして発行する「SPI 活動レポート」や個別フィードバックに載せる図や表を自動的に生成できる．個別フィードバックの目次を図 4.12 に示す．

個別フィードバックでは，過去に提出されたデータの履歴も利用し，経年での QCD の指標の推移も提示している．この推移から，SPI 活動とその効果の関係が確認できるようになっている．また，SPI 活動レポートで示した全社の分析結果の中で，自部門がどのあたりにプロットされるのかを示している．このような結果を参照することで，全体の中で

図 4.11: SPI 活動レポートの目次

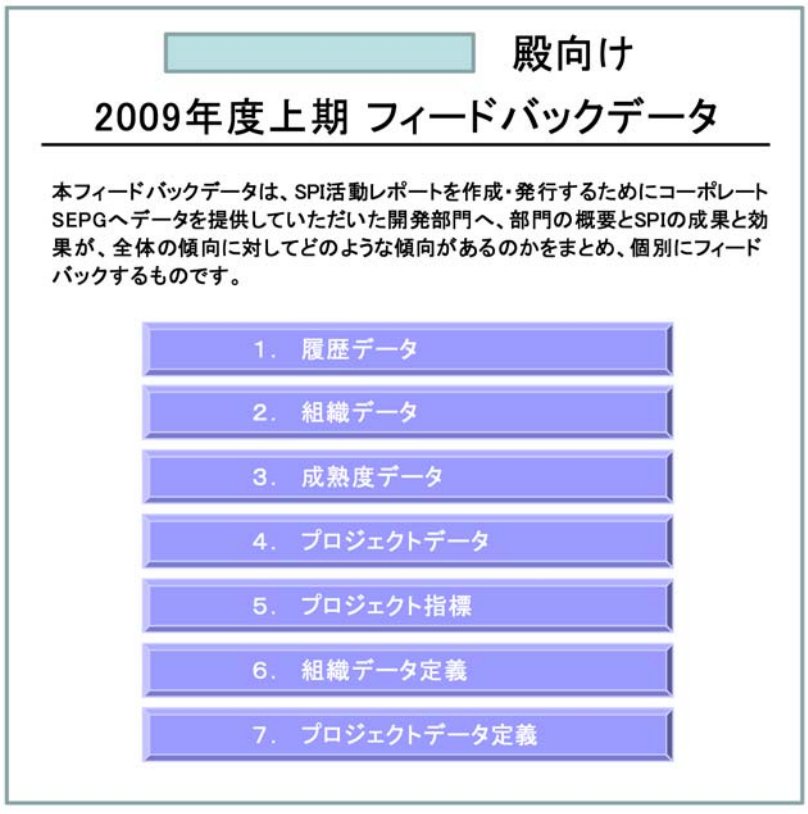


図 4.12: 個別フィードバックの目次

の自部門の位置付けが把握できるようになる．

4.5　改善活動の経験や事業領域の違いによる SPI 活動の進め方

SPI 活動の進め方としては，大きく 3 つに分けることができる [60].

1 つめのグループは，SPI 活動を開始して 4 年以上継続しているか，元々伝統的に SPI 活動を継続してきた組織であり，レベル 3 以上の成熟度になっている組織である．当然 SEPG，SQAG が確立されているため，日常的なさまざまな課題はあるが，それをインプットに SPI 活動を継続している組織であり，ISO9001 との調整もうまくいっている場合が多い．

2 つめのグループは，SPI 活動を開始して，レベル 2 前後に成熟している組織である．順調に進んでいる場合もあるが，途中から SEPG の工数が取れずに，レベル 2 から進んでいない組織もある．プロジェクト主体のレベル 2 から，組織主体のレベル 3 になるためには，スポンサーの強いコミットメントが不可欠であるが，進みが止まっている組織では，そこに苦慮している場合が多い．外部からの刺激が必要である．

3 つめのグループは，これから SPI 活動に着手する組織である．組織の現状の課題を解決するために，新たに SPI のネットワークに入ってきた組織であり，SEPG を立ち上げるところである．診断と改善という従来の SPI 活動スタイルとは違って，ワークショップやヒアリングなど，イベントを中心に，まずは組織内のコミュニケーションの活性化から着手する組織も出現し，SPI 活動の新しいやり方として注目されている．

上記のような現状の活動状況から,「改善モデル型」の改善と「課題解決型」の改善の違いについて整理する．

「改善モデル型」の改善とは CMM や ISO9001 のように，ある標準やデファクトを参照しながら，自分たちの強い点弱い点を診断し，強い点をより強く，弱い点を強くするために改善活動を進めるやり方である．モデルがあるため，場当たり的になりづらく，目標もはっきりするため，中長期的な改善活動には向いている．場合によっては，SEPG や SQAG，トレーニンググループなどの組織化も要求されてくるため，大規模な組織が取り組みやすい．ただ規定や標準といったルール文書主義になりやすく，ISO 取得やレベル達成といった，手段が目的化する活動に変質する恐れがあり，注意を要する．

一方,「課題解決型」の改善とは，その組織の課題をまずは洗い出し，その優先順位の高いものから改善を進めるというやり方である．組織が改善して欲しいというものから改善を始めるため，効果も納得されやすく，改善に着手しやすい．またこの方法は，目的を絞ったツールの活用，例えば構成管理ツールやプログラム静的解析ツールなどの活用が有効の場合に，特に効果がある．小規模組織が取り組んでいる場合が多い．しかし改善アイテムにバラツキがあり，場当たり的な改善になってしまう場合がある．例えば，CMM レ

ベル 2 の構成管理が十分できていないにも関わらず，追跡管理（トレーサビリティ）を組織の中で取り込もうとすることは非常に難しい．簡単なレベルからまずは始めることが肝要となる．

東芝グループでは，社会インフラ系に「改善モデル型」の改善が多く，組み込みソフトウェア系に「課題解決型」の改善に取り組む組織が多い．しかし最近は，CMMI 連続モデルの活用により,「課題解決型」の改善の中で，CMMI のプロセス領域を活用して，診断し，改善をするという新しい改善方法の流れが出てきている．これは「改善モデル型」の改善と「課題解決型」の改善の融合の 1 つの例である．

4.6 全社的な SPI 活動実践による効果

本節では，全社的な SPI 活動実践の結果として，組織成熟度の達成状況と QCD の効果について示す．さらに，それらの結果を基に，以下に示す本活動の目的に対する達成度を評価する．

- **目的**
  - ソフト生産技術力の向上
  - 改善のできる組織文化の構築

4.6.1 ソフト生産技術力の向上 ... QCD の効果

SPI 活動レポートを作成する際に収集したデータをもとに，開発期間，開発工数，開発規模について，半期（半年）毎のデータをプロットして，傾向を確認している．図 4.13 に，開発工数の予定と実績差をプロットした結果を示す．成熟度別に分類せず収集したデータをすべてプロットしているため，このグラフからは，バラツキの幅が小さくなっているという顕著な傾向は確認できていない．しかしながら，全体的な傾向としては，予実差が縮小傾向にあることは確認できる．

2009 年度上期に発行した SPI 活動レポートでは，図 4.13 に対して以下のコメントを付記している．

- 平均値，中央値，第 1 四分点と第 3 四分点の範囲は，先期とほぼ同じ傾向である
- 全体的には，多くのプロジェクトが予定と実績に大きな乖離がなく開発ができている

一方，組織の成熟度の向上に伴い「定量的プロジェクト管理」が実績できる組織が増えてきている．成熟度の高い組織では，ソフトウェア開発における精度の高いデータが収集できるようになり，そのデータを分析して，開発の節目節目でフィードバックするという仕組みが確立できている組織も増えてきた．ここでは，以下の 2 つの事例を紹介する．

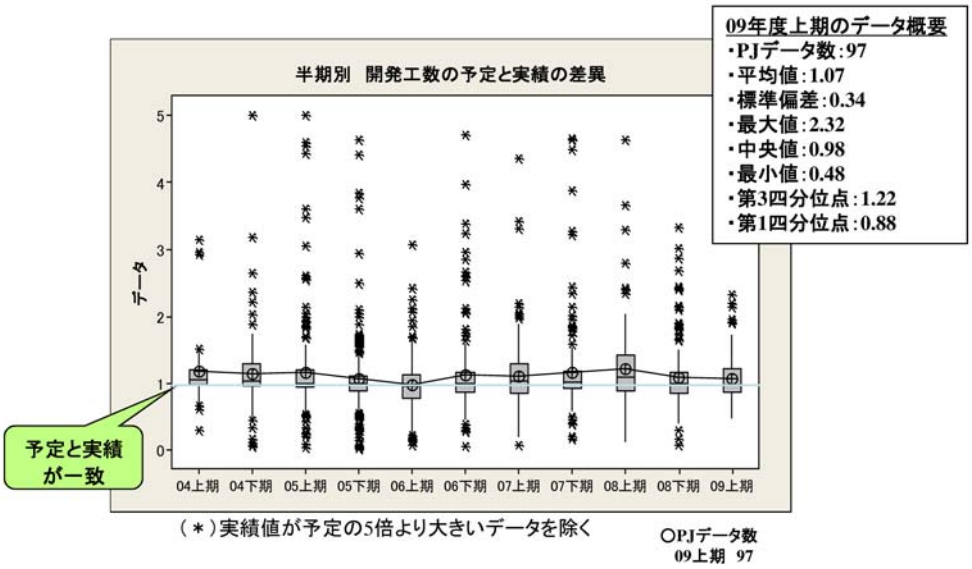


図 4.13: 開発工数の予定と実績の差違

- 事例 1：SPI 活動における QCD の効果
- 事例 2：定量的プロジェクト管理における不具合数の予測

事例 1：SPI 活動における QCD の効果

この組織では，成果物ボリュームの予定と実績，作業毎の工数予定と実績，レビュー／検査で検出できる不具合数の見込みと実績および不具合混入工程のデータを，プロジェクトの進行中から図 4.14 に示したようなグラフで可視化している．プロジェクトのマイルストーンでこの結果を確認し，もし，予実差に大きな乖離がある場合には何らかの対応を取るというプロセスを実践している [63].

図 4.14 のような定量的なプロジェクト管理ができている組織では，以下のような QCD 指標を SPI 活動の効果として示しているところが多い．

- 納期確保（初期納期ずれ）
- 見積り精度（初期工数ずれ）
- 上流検出率（上流指向）
- 下流バグ密度（製品品質）

図 4.15 に，QCD 指標の状況を半期（半年）毎にプロットした結果を示す．

図 4.15 の上流検出率・下流バグ密度を示した図の左型の縦軸は上流検出率を表し，右側の縦軸は下流バグ密度を表している．この図の上段のグラフからは，納期と工数の見積りと実績の差（ずれ）が年々小さくなっているということが分かる．また，下段のグラフからは，上流検出率の向上に伴い下流バグ密度が小さくなっているということが分かる．

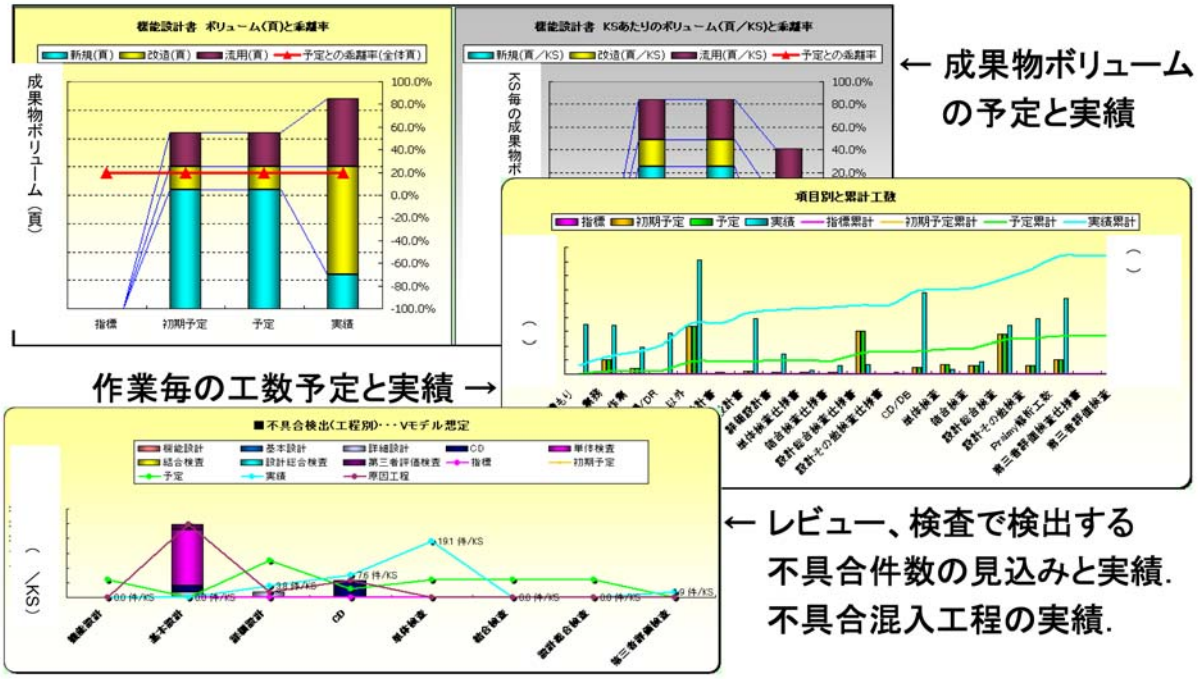


図 4.14: 規模，工数，不具合数の予定と実績の可視化

事例 2：**定量的プロジェクト管理における不具合数の予測**

この事例では，システムテストにおける不具合数を予測するためのプロセス実績モデル（PPM:Process Performance Model）を構築している [63][64].

PPM を構築するために，まず過去のプロジェクトデータのスクリーニングを行い，システムテストの不具合密度が平均から 1 σ 以内のプロジェクトを対象データとする．次に回帰分析を行い,「新規／改造コード行数」を説明変数として,「総不具合数」を従属変数とする相関があることが分かった（図 4.16 参照）.

この回帰式から，プロジェクト開始時に新規／改造コード行数を見積りできれば，総不具合数の予測を得ることができる．しかしながら，このままでは総数しか分からないのでプロジェクト管理にうまく活用することができない.

そこで，次にこの総不具合数を各フェーズに振り分けることを行う．過去プロジェクトの実績データに基づいて，各フェーズの発生割合（平均）を求めることで，フェーズ毎の不具合プロファイルを得ることができる．さらに，フェーズ毎の発生件数の 75%を上限，25%を下限として制御するための範囲を設定する．つまり，過去の 50%がこの範囲内に入っていることを意味している（図 4.17 参照）.

実際のプロジェクト管理では，この上下限を超えた時に何らかの異常が発生していると判断して，適切なアクションを取ることになる.

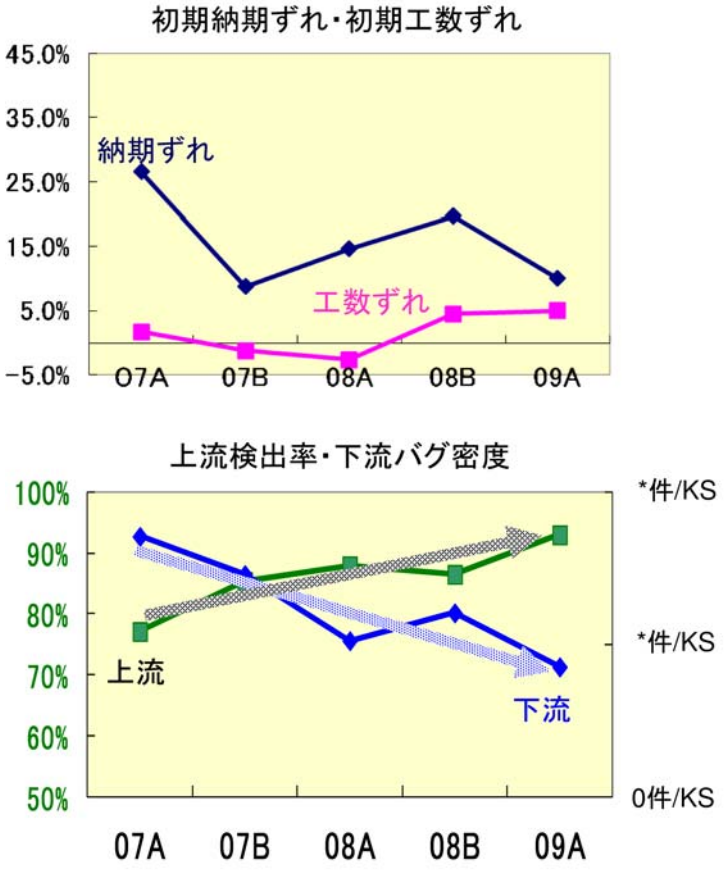


図 4.15: QCD 指標の状況

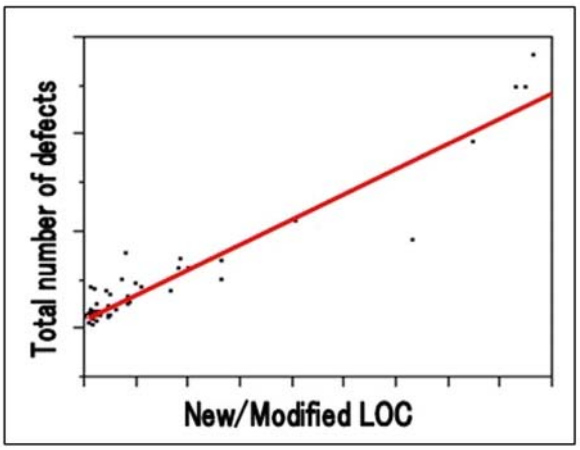


図 4.16: 回帰分析：新規・改造コード行数と総不具合数

この事例では，定量的なデータに基づいて作成された PPM が実際に活用されていることがポイントである．実際に役立つ予測を行うために，プロジェクトの特徴の類似性に基づいて収集データを分類すること，成功したプロジェクトのスクリーニングを行うこと，が

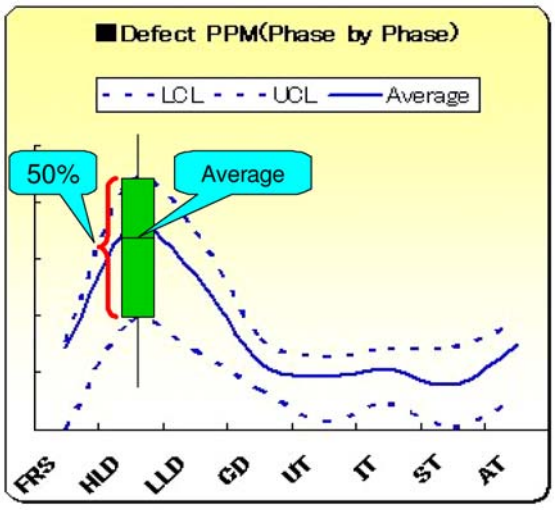


図 4.17: 不具合件数プロファイル

重要なポイントであり，さらに加えて，上下限（UCL:Upper Control Limit，LCL:Lower Control Limit）を超えた時に「通常とは違う何か」が起こっていることを適確に捉え，より詳細な分析を行ったうえで適切なアクションを取っていくことも実際のプロジェクト管理では重要である．

4.6.2 改善のできる組織文化の構築 … 定着と成熟度の向上

図 4.2 の SEPG 設立部門数の推移で示したとおり，SPI 活動推進のための 3 階層 SEPG の体制が構築され維持されていることが分かる．また，この開発部門 SEPG の数は約 60 部門となっている．この数は，活動開始時点で SEPG の設置が必要と考えた部門数の約 80%に達している．

2005 年以降は，カンパニー SEPG が，コーポレート SEPG と協力して開発部門 SEPG を支援できるようになってきている．2008 年時点では，支援対象のうちの約 75%の開発部門に対してカンパニー SEPG が支援をしており[5]，開発部門に対する支援の中心が，コーポレート SEPG からカンパニー SEPG に推移している．表 4.2 で示したように，当初から定着フェーズではコーポレート SEPG のリソースを減らす計画をたてていたが，開発部門の自立化が進んだことと，カンパニー SEPG の整備が進んだことで，計画どおりのリソースで推進できることを示せた．

次に，組織の成熟度の状況を確認する．

図 4.18 は，2000 年からの社内公式アセスメントの結果の累計を年度毎に積み上げたものである．SPI 活動を推進している約半分の開発部門が，この社内公式アセスメントを利

[5]コーポレート SEPG と一緒に支援している部門を含む

用している．

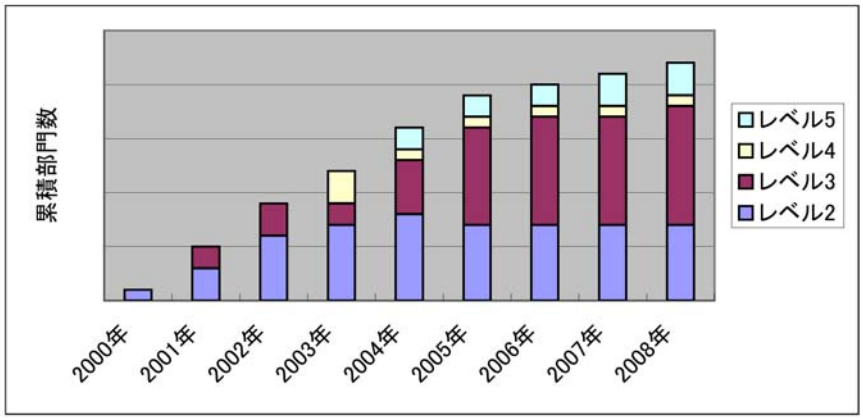


図 4.18: 組織成熟度の達成状況

図 4.18 からも分かるとおり，レベル 3 以上の成熟度を達成した開発部門が年々増加している．このことから，開発部門における成熟度は着実に向上していると判断できる．また，社内公式アセスメントを利用していない部門でも，SPI 活動レポート作成のために収集するデータから，開発プロセスが着実に整備されているという実績が得られている．

また，SPI 活動が浸透し，SPI に関連する技術の習得が進むことによって，いろいろな改善活動の方法が提案されてきた．文献 [57, Iida] では，CMMI 連続モデルを活用し，課題ベース改善の中で診断を行い，その結果に基づいて改善項目を抽出するという方法が提案されている．また，文献 [48, Aoki] では，小規模の開発部門において，ワークショップをベースにした SPI 活動の具体的な推進方法と効果が述べられている．

第5章　考察

本章では，2000 年から現在も継続している全社的な SPI 活動について，以下の 8 つの観点から考察する．

- 東芝版 SPI フレームワークの構築手順
- 各フェーズにおける工夫点
- 東芝版 SPI フレームワーク構築から得られた知見
- SPI フレームワークを構成する要素間の関係
- SPI フレームワークの実現可能性
- 提案した SPI フレームワークの汎用性
- 東芝版 SPI フレームワークの実用性
- 10 年間にわたる全社的 SPI 活動の総括

5.1　東芝版 SPI フレームワークの構築手順

SPI フレームワークを構成する 6 つの活動を，全社的な SPI 活動開始時点から充実したものに仕上げておくことは現実的には難しい．そこで，それぞれの活動をどのように充実したものにしていくのかという計画を立案した（表 5.1）．

2000 年から表 5.1 で示したステップで，SPI フレームワークを構成する 6 つの項目に関する具体化を進め，計画，実装フェーズの最初の 4 年間で，全社的な SPI 活動推進のための仕組み（東芝版 SPI フレームワーク）を構築した．その後，この枠組みを変えることなく，個々の内容を充実させながら全社レベルでの SPI 活動を継続している．

表 5.1 で示した計画は，SPI フレームワークを構成する各構成要素に対するものである．各構成要素を構築する順番と全社的な SPI 活動の制度化（全社規定などを制定し，全社的な活動を確実なものとするための仕組み）のタイミングを，図 5.1 に示す．

活動の初期の段階では，コア技術を開発し，それを自分達のものとしたうえで試行評価することが最も重要なポイントである．その後，SPI 活動を推進している開発部門に対するサービスメニューを整備したうえで，コア技術とサービスメニューを継続して改善し続ける．SPI 活動が全社的な広がりを見せた時点で，制度化することが効果的と考えている．さらに，SPI 活動自体には終わりはないため，定着フェーズでは，次の 10 年（期間は全

表 5.1: SPI フレームワークを構成する各項目の推進計画

		推進フェーズと年度			
		計画	実施（基礎固め）	展開	定着
		2000–2002	2003–2004	2005–2007	2008–2010
強化すべき6つの活動の具体化手順	活動1:「改善のモデル」の選択と習得	改善のモデル（ロードマップ、改善のサイクル）選択とモデルの習得	モデルの理解度を高め、推進部門としての知識化・資産化を推進		次のサイクルで活用するモデルの調査・研究、既存モデルの再整理
	活動2:「推進体制をベースとしたSPI活動」の推進	全社的なSPI活動推進のための最適な推進体制の検討	全社的なSPI活動推進体制の構築	全社的なSPI活動推進体制の維持と発展	
	活動3:「改善の技術とスキル」の習得	SPI活動推進に関する技術要素の抽出と獲得	SPI活動を推進するための方法論やツールの開発・適用と，それらの方法論やツールを習得するためのトレーニングコースの開発・展開		左記に示した項目の改善と運用
	活動4:「管理手法／管理ツール」の選択と導入・展開	管理手法／管理ツールの選定とソリューションの構築	管理手法／管理ツールの全社的な推進体制の構築	利用者間のネットワークの確立と利用部門からのフィードバックに基づく改善。新規に導入する管理手法や管理ツールの選定。	
	活動5:「プロセス改善活動推進のための情報基盤」の確立	情報基盤の種類と目的の明確化	情報基盤として提供する取り組みの先行評価と情報基盤の構築	情報基盤の確立	情報基盤を利用した定期的な情報の発信と蓄積
	活動6:「改善活動の成果と効果」の可視化		改善活動の成果と効果の可視化に対する基本的な考え方の確立	収集データと分析方法の決定と先行部門での適用評価	メトリクスプログラムの確立と展開

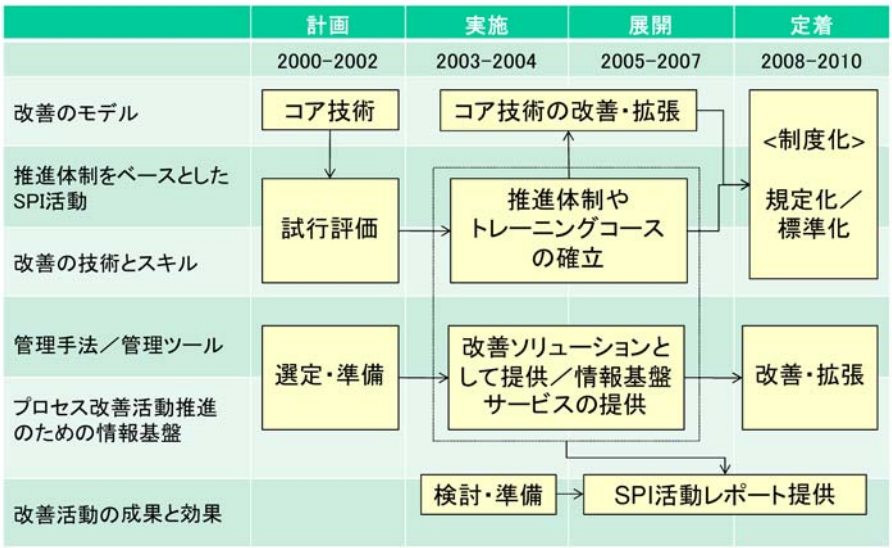


図 5.1: SPI フレームワークの構成要素を構築する順番と制度化のタイミング

社的なSPI活動の計画によってそれぞれである）に向けての準備と計画立案に着手することが必要となる．

5.2 各フェーズにおける工夫点

ここでは，表4.2，表5.1で示した，計画，実施（基礎固め），展開，定着の4つのフェーズで工夫したことを述べる．

5.2.1 計画フェーズ

このフェーズで注力したことは，以下の3つである．

- 経営層のコミットメントを確実なものとする
- 各カンパニーのスポンサーからの協力を確実なものとする
- 推進メンバの技術力を向上させる

経営層からのコミットメントを確実なものとするために，経営層に対する報告会を定期的に開催し，経営視点からのコメントを適切に反映することを心がけた．このような取り組みによりコミットメントが確実なものとなり，アセスメント費用や追加のリソースの確保ができた．

各カンパニーのスポンサーからの協力を確実なものとするために，スポンサーとの話し合いの機会を定期的に作った．この時，形式的な議論にならないように，本音で議論することを心がけた．この時期に，主要なカンパニーのスポンサーと十分に話し合いができたことで，このフェーズ以降の活動に対する協力が確実なものとなった．

全社レベルでのSPI活動を推進するために，開発部門を支援するメンバの育成と，支援する際に必要となる道具（武器）の開発が重要となる．支援メンバの育成のために，この活動のコアメンバとなる6名のリードアセッサを育成した．この6名のコアメンバを中心に，アセスメントを複数の部門で実施し，CMMの活用方法やSPI活動の推進方法に関する技術とノウハウを蓄積した．さらに，CMMをより良く理解するために，コーポレートSEPGとカンパニーSEPGメンバが共同で，プロセスエリア毎のガイドブックを作成した．このガイドブックには，社内の事例などを盛り込み，できるだけ実践的な内容になるように心がけた．また，改善のサイクルとしてPDCAという考え方はよく使われるが，具体的にどのように回せばよいのかは示されていないため，IDEALをベースにして，具体的なSPI活動の推進ガイドを作成した．

5.2.2 実施（基礎固め）フェーズ

このフェーズでは，支援部門数が急増した．開発部門に対しては，必ず二人ペアで支援するという方針をたてた．また，支援メンバの SPI に関連する技術とスキルを向上し，ノウハウを共有するため，毎週一回は全員が集まる日とし，その日に，推進部門として実施すべきさまざまな活動を集約して実施した．また，この期間には，支援活動，教育，社内イベントに関するさまざまな成果物が出てくる．そこで，成果物を管理するとともに，社内への発信を強化するために専任のメンバ（バックオフィス）を置いた．

5.2.3 展開フェーズ

各開発部門における SPI 活動の自立化を促進するため，カンパニー SEPG との連携を強化し，コーポレート SEPG とカンパニー SEPG メンバで開発部門を支援するという体制を確立した．また，SPI 活動の進め方に関するバリエーションを増やすため，今までの実践結果を整理した（ワークショップをベースにしたプロセス改善活動 [53]，改善文化を創る SPI〜改善意識中心アプローチ〜[54] など）．

さらに，全社レベルの活動を確実なものとするために,「ソフトウェア開発ガイドライン 〜ソフトウェアプロセス改善 (SPI) 推進ガイド〜」と「東芝グループ ソフトウェア品質保証（ガイドライン)」の 2 つのガイドラインを制定し発行した（詳細は，4.4.1 節を参照のこと）．

5.2.4 定着フェーズ

各開発部門の成熟度が向上し，より効果的・効率的な手法やツールの導入を求める声が多くなってきた．そこで，いくつかの部門で効果のあった品質管理手法／ツールを活用した SPI 活動を推進するために，品質管理技術を担う役割をコーポレート SEPG に追加した．また，カンパニー SEPG メンバとコーポレート SEPG メンバとのローテーションの仕組みを確立した．

5.3 東芝版 SPI フレームワークの構築から得られた知見

ここでは，東芝版 SPI フレームワークを構築する際に得られた知見を構成要素毎に，成功要因，工夫点，効果としてまとめた．

5.3.1 “活動 1 :「改善のモデル」の選択と習得 ”から得られた知見

CMM，IDEAL ともに，ただ単にモデルをそのまま利用するのではなく，東芝グループの事例を交えたガイド（全キープロセスエリアのガイド，IDEAL ガイド）を，初期のフェーズで作成した．推進部門側にとって，”自分たちの武器”を持つことが，以後の活動に対して大きな効果を発揮した．さらに，モデルの理解を深めるため，コアメンバをリードアセッサ／リードアプレイザとして育成した．専門家を複数人育成できたことで，専門家集団として認知してもらえるようになった．

- 成功要因
  - 成熟度モデルとして完成度の高いモデル CMM を選択
  - SPI 活動のステップとして IDEAL を選択
- 工夫点
  - CMM，IDEAL ともに，社内の事例を交えたガイド（全キープロセスエリアのガイド，IDEAL ガイド）を作成した
  - これらのガイドは，SPI 活動を推進するコーポレート SEPG やカンパニー SEPG メンバにとって，開発部門を支援するための強力な道具となった．また，モデルの理解を深めるため，コアメンバの数名をアセッサとすることで専門家を育成した
- 効果
  - 完成度が高いモデルやステップだったので，それらを理解するための文献や情報を数多く入手することができた
  - 社内の事例を交えたガイドを作ることで，モデルを自分たちの言葉で説明できるようになった
  - これらのガイドは，コンサルティングやトレーニングの核となった
  - アセッサを育成したことで，専門家として認知してもらえるようになった

5.3.2 “活動 2 :「SPI 活動の推進体制」の構築 ”から得られた知見

開発部門の SPI 活動を支援する人的リソースについて述べる．表 4.2 に示したとおり，展開フェーズにおける支援部門数の増大への対応が必要である．展開フェーズで必要となる人的リソースを以下の計算式から算出した．この計算式は，計画，実施フェーズの結果から導出したものである．

$$\text{必要な人的リソース} = (\text{支援部門数} \div (\text{担当者一人当たりの支援部門数} \times \text{週を単位としたサポート間隔})) \times \text{一部門を支援する人数}$$

展開フェーズで同時期に支援すべき開発部門数が 30，担当者一人当たりの支援部門数が 2，一つの部門を 2 名で支援し，2 週間に 1 回のペースで対応（改善内容の検討を行う SEPG 会議への参加や改善活動に対するコンサルティング）する場合には，(30 / (2 * 2)) * 2 となり，カンパニー SEPG とコーポレート SEPG を合わせて，15 名のメンバが必要となる．

- 成功要因
  - 改善活動を推進するグループとして SEPG を定義し，各階層に SEPG を設置
- 工夫点
  - 3 階層で SEPG を設置（適切な階層に推進部門を配置）
  - 全社活動の初期段階で，カンパニー SEPG メンバと一緒に，ガイドやトレーニングコースなどを開発（早期に技術の底上げを図る）
  - 開発部門の SPI 活動を，カンパニー SEPG とコーポレート SEPG でサポート（きめ細かな支援）
- 効果
  - カンパニーの方針や計画を SPI 活動に反映しやすい
  - 組織文化や製品開発の特徴をよく理解したうえで SPI 活動を推進できる
  - 開発部門に改善活動に責任を持つグループを設置することで，プロセスオーナーシップの意識が高くなる

5.3.3 “活動 3：「改善の技術とスキル」の習得”から得られた知見

全社的な SPI 活動の初期の段階から「SEPG リーダコース」を開設し，リーダ育成を積極的に実施した．この当時（現在も続いているが），全社的にシックスシグマ活動も推進しており，この活動の中で提供されているシックスシグマ活動推進者を育成する教育の枠組みを参考に，SEPG リーダコースのカリキュラムを決めた．その後，SQAG 関連のトレーミングコースを提供したことも，各開発部門における SPI 活動の実践をより強固なものにしている．

また，開発部門の SPI 活動を支援する推進側での改善活動も大切なポイントとなる．開発部門におけるプロジェクトではソフトウェアを開発し，SPI 活動を推進する部門側のプロジェクトでは，SPI 推進のためのさまざまなコンテンツや管理手法／管理ツールなどの研究・開発を進める．開発するものは違うが，プロジェクトとして計画をたて，進捗管理をし，開発の途中で成果物のレビューをするという，プロセスの観点では同じ作業をしている．つまり，支援部門側の組織成熟度が低ければ，それ相応の支援しかできないことは自明である．そのため，推進部門側のプロセス改善をしっかりと進めることが大切である．

- 成功要因
  - トレーニングの提供と推進部門側の組織的活動
  - SEPG と SQAG のためのトレーニング提供
  - 推進部門側の改善活動
- 工夫点
  - SEPG と SQAG のためのトレーニングコースを開発し提供（開発部門の SPI 活動推進のためのスキルの明確化とその向上）
  - 支援活動は，2 人でチームを組み実施（偏りのない支援を提供）
  - 支援内容（結果，工夫点，失敗点など）を組織として定期的にまとめ共有化（自分の経験以外の支援技術の獲得）
- 効果
  - 開発部門における SPI 活動関連メンバのスキルが向上
  - 支援部門側でのプロセス改善活動をとおして組織メンバの経験や成果が共有・蓄積されることで，提供するトレーニングや支援活動の品質が向上する

5.3.4 “活動 4：「管理手法／管理ツール」の展開”から得られた知見

コーポレート SEPG 設立時は，人数も少なく，管理手法／管理ツールの調査・研究，導入支援を担当するメンバを割り当てることは難しかった．これらの手法やツールの導入による改善効果は非常に高いため，この領域に対するリソースを強化するとともに，コーポレート SEPG メンバの個々のメンバが得意とする専門領域を作ることに心がけた．

- 成功要因
  - 効果が高い管理手法／管理ツールを適切に選択
  - コーポレート SEPG 主導で技術開発，ツール導入
  - 情報共有できる場の設定
- 工夫点
  - ソフトウェア開発における品質向上に効果の高い高額な手法やツールを開発部門で導入するには負担が大きいため，コーポレート SEPG がツールベンダと一括契約し，利用する部門は，契約で締結した使用状況に合わせて利用料をコーポレート SEPG に支払うというモデルを確立
  - ユーザ会を設置し，定期的に開催．最初はツールの機能紹介や効果的な利用方法に関する話題が多いが，利用期間が長くなると，そのツールで収集できるデータの分析などに興味が移ってきた．そこで，年 2 回開催するユーザ会で，1 回

目はツールの機能紹介を中心に実施し，2 回目はデータ分析を中心に実施するように変更

- 効果
  - 管理手法／管理ツールを適切に提供できる
  - 課題解決型のアプローチにおける改善活動の立ち上げ時期に活用することで，1 サイクル目の改善活動の成功率が高まる

5.3.5 “活動 5：「SPI 活動推進のための情報基盤」の確立”から得られた知見

イベントを開始した当初は，社外からネームバリューのある方を招待し，経営層の方やソフトウェア開発部門の部長などに講演を聞いてもらう機会を意識的に作った．イベントの会場では，参加者間のコミュニケーションを図る場として，展示コーナーなどを準備した．また，展示コーナーには，コーヒーやお菓子なども置き，リラックスして過ごせる空間を作ることを心がけた．

- 成功要因
  - 参加してもらいたい参加者層の特定
  - 参加者層ごとに提供するイベントやインフラ環境を整備
- 工夫点
  - ソフトウェアフォーラム（2 日間），SEPG ワークショップ（1 日間），SEPG リーダフォローアップコース（1 日間）を開催（SPI 活動推進担当者がモチベーションを維持・向上してもらうための場）
  - SQAG のための教育として SQAG リーダコース（6 日間），SQAG 入門コース（2 日間）を開催
  - 支援を継続するというメッセージを伝えるために定期的に開催（安心感，信頼感の提供）
  - コーポレートの SPI 推進施策を検討し，実施状況を確認する場として SPI 委員会を設置（第三者的な観点からの検証の場）
- 効果
  - 社内の人的ネットワークが広がるとともに，強固になる
  - SPI 活動の推進結果を発表する場として活用できる

5.3.6 “活動 6 :「改善活動の成果と効果」の可視化 ”から得られた知見

開発部門からのデータ提供後，できるだけ早く SPI 活動レポートと各開発部門へのフェードバックレポートを発行するように心がけた.

- 成功要因
  - SPI 活動の成果と効果を定義
  - 定期的に SPI 活動レポートを発行し，それぞれの階層で状況を報告
- 工夫点
  - CMU/SEI から毎年発行されている Maturity Profile を参考に，SPI 活動レポートの発行を企画し実行
  - SPI 活動の成果や効果の考え方を提示する
    - 成果：定着状況や組織成熟度の達成状況
    - 効果：QCD に関連するデータを活用
  - データ提供への心理的負担を軽減するために，収集したデータの活用方法を提示
    - 第三者には生のデータを開示しない，組織を横並びで評価しない，SPI 活動レポートでは絶対値の指標を出さない
    - 個別フィードバックでは，QCD の年度推移を提示（SPI 活動の効果を定量的に示す
- 効果
  - 開発部門において，SPI 活動の成果と効果を示すためのデータの収集と分析のプロセスが確立できる
  - 3 階層 SEPG の各層で，成果と効果を示すことができる

5.4 SPI フレームワークを構成する要素間の関係

5.1 節では，SPI フレームワークを構成する要素をどのように組み上げていけばよいのかを示した．ここでは，SPI 活動を推進・定着させるために提案した「SPI フレームワーク」に基づいて構築した東芝版 SPI フレームワークにおいて，構成要素間の関係が特に強い項目について説明する．これらの関係を理解しておくことで，構築した SPI フレームワークを効果的に運用することが可能となる．構成要素間の関係が特に強いものを図 5.2 に示す.

SPI フレームワークを構成する要素間で特に関係の強いもの（図中の①〜③）を以下に示す.

- ①：活動 1 と活動 3

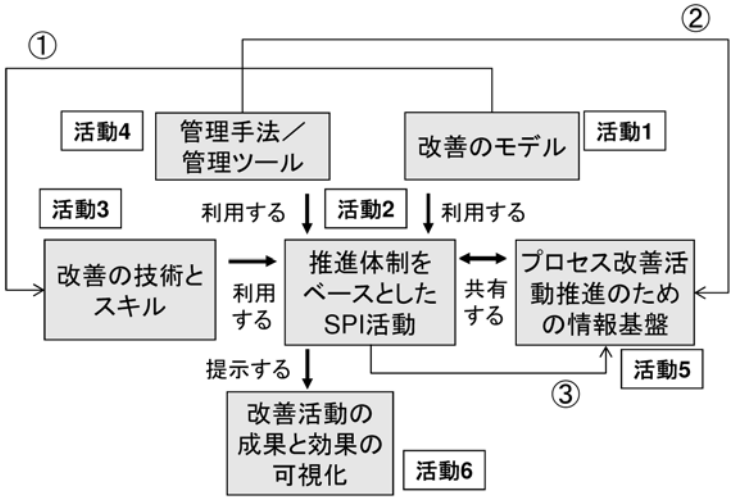


図 5.2: SPI フレームワークを構成する構成要素間の関係

  - 活動 1 では，改善のモデルとして採用した CMM や IDEAL に関するガイドを開発する．これらのガイドは，トレーニングで活用される．このガイドの品質がトレーニングの品質に大きく関係する．また，このガイドは，活動 2 の SPI 活動でも利用される．

- ②：活動 4 と活動 5
  - 活動 4 では，コーポレート SEPG やカンパニー SEPG が管理手法や管理ツールを改善ソリューションとして整備する．これらの手法やツールを広く展開するには，まずは多くの方々に認知してもらうことが重要となる．活動 5 の情報基盤を活用することで，認知度を高め，広く展開するための情報発信が可能となる．

- ③：活動 2 と活動 5
  - 各開発部門の SPI 活動自体は，活動 2 で構築した体制を活用して実践されている．SPI 活動の実施結果をイベントなどで発表することで，情報基盤に蓄積される情報がより豊富になる．さらに，SPI 活動の実施状況を定期的に SPI 委員会で確認することで，全社レベルでの施策検討に役立てることができる．

5.5 SPI フレームワークの実現可能性

東芝版 SPI フレームワークを構成する各活動の内容を決める際に参考としたもの，主要な成果物／活動のレベル，他社と比較した場合の優位性についてまとめた結果を表 5.2 に示す．

表 5.2: 各活動における実施内容と評価結果

構成要素	実現しようとしたこと	主要な成果物／活動	参考にしたもの	主要な成果物／活動のレベル			他社と比較した優位性
				新規性が高い	既存の改善	一般に提供されている	
活動1:「改善のモデル」の選択と習得	CMMIと改善のサイクルをより良く理解したい	社内の事例を盛り込んだ，キープロセスエリアA毎のガイドブック(それぞれ50～70ページ)	CMMに関する参考図書		○		○
		改善のサイクルであるIDEALの具体的な進め方や留意点などを整理してガイドブックを作成	CMU/SEIから発行されているIDEALに関する報告書		○		○
活動2:「プロセス改善活動」の推進体制	開発部門との連携を強化し，より実践的な支援を提供したい	3階層SEPG	過去の支援経験(カンパニーレベルの技術支援スタッフを巻き込むと活動がスムーズに進められることが多かった)		○		○
活動3:「改善の技術とスキル」の習得	開発部門におけるSPI活動をより効果的・効率的に推進したい	開発部門のSPI活動を支援するためのメンバと推進方法	コンサルティング会社によるコンサルティング方法			○	
		全社標準のソフトウェア開発プロセス	CMMIに準拠したプロセス一式		○		
		SEPGリーダコース	経営変革手法「シックスシグマ」で提供されているトレーニングの構成	○			○
		SQAG入門コース／リーダコース	自社で開発したSEPGリーダコースの構成と実施経験	○			○
		CMMI関連トレーニング	CMU/SEIから提供されてるCMM関連のトレーニングコース			○	
		プロセス診断手法	CMM/SEIから提供されているアセスメント／アプレイザル手法		○		
活動4:「管理手法／管理ツール」の展開	品質向上に役立つ管理手法や管理ツールをタイムリーに提供したい	管理手法／管理ツールを導入・定着させるための改善ソリューション	ツールベンダの説明方法や導入サポート	○（手法／ツールの機能）	○（導入・定着の方法）		○
活動5:「プロセス改善活動推進のための情報基盤」の確立	人と人のネットワークを構築し，さまざまな情報をタイムリーに発信するとともに，自社内の資産を有効に活用したい	イベント(ソフトウェアフォーラム，SEPGワークショップ，SEPGリーダフォローアップ)	日科技連やソフトウェア技術者協会主催のシンポジウム			○	○
		情報発信／共有(掲示板，ニューズレター，メーリングリスト，ユーザ会)	さまざまな社外のコミュニティ活動やメールニュース			○	
		SPI委員会	社内における全社的な活動を推進するための体制			○	
活動6:「改善活動の成果と効果」の可視化	各階層でSPI活動の成果と効果が可視化でき，次のSPI活動に活用してもらいたい	SPI活動レポート	CMU/SEIから定期的に発行されているMaturity profile			○	

表 5.2 に示したとおり，東芝版 SPI フレームワークとして新規性が高く，他社と比較して特に優位性の高いものは，以下の 3 つと考えている．

- SEPG リーダ向けのトレーニングである「SEPG リーダコース」
- SQAG 関連のトレーニングである「SQAG 入門コース／リーダコース」
- 管理手法／管理ツールを導入・定着させるための改善ソリューション

上記のトレーニングコースについては，コーポレート SEPG とカンパニー SEPG が中心となり改善／改訂を繰り返すことで，毎年より良いトレーニングを提供できるようになってきている．継続して提供することが重要なポイントである．また，管理手法／管理ツールについては，調査や研究・開発の段階ではリソースが必要となるため，手法やツールの選択が大切なポイントである．社内外から情報を収集し，適切なものが選べるようにしておく必要がある．コーポレート SEPG やカンパニー SEPG が，管理手法／管理ツールの研究・開発に対してミッションがない場合には，ソフトウェア生産技術を担っている関連部門などと連携を取ることが必要となる．

上記以外のものは，既存のものを改善したレベル，一般に提供されているレベルのものである．つまり，SPI フレームワークの構成要素の内容を決めて，それを準備すること自体はそれほど困難なことではないことが分かる．このことは，本研究で提案した SPI フレームワークに基づいて，自部門用の SPI フレームワークの構築が実現可能であることを示している．

5.6 提案した SPI フレームワークの汎用性

ここでは，CMMI で提示されている概念を用いて，提案した SPI フレームワークの汎用性を確認する．SPI フレームワークでは以下の 6 つの活動を構成要素として配置している（具体的な内容については，3 章を参照のこと）．

- **活動 1**：改善活動を実施する能力
- **活動 2**：推進体制を構築して改善活動を実践
- **活動 3**：改善活動の進め方に関する能力
- **活動 4**：管理手法／管理ツールを調査して選定する能力
- **活動 5**：改善事例の提供と他部門の改善事例の利用
- **活動 6**：改善活動の成果を論理的に説明

CMMI のプロセス領域は,「プロセス管理」,「プロジェクト管理」,「エンジニアリング」,「支援」の四つの区分にグループ化できる．

さらに，プロセス管理は，基盤「プロセス管理のプロセス領域」と累進「プロセス管理のプロセス領域」に分けられる．基盤「プロセス管理のプロセス領域」は，ベストプラクティス，組織プロセス資産，および学習結果を文書化し，組織横断的に共有するための能力を組織に提供する．また，累進「プロセス管理のプロセス領域」は，組織の「品質およびプロセス実績の定量的目標管理」を達成するために，改善された能力を組織に提供する．

このように，SPI フレームワークで提案している 6 つの構成要素は，CMMI の「プロセス管理のプロセス領域」と密接に関係していることが分かる．

「プロセス管理のプロセス領域」に含まれるプロセスと SPI フレームワークの構成要素間の関係をまとめた結果を表 5.3 に示す.

表 5.3: プロセス管理のプロセス領域と SPI フレームワークの構成要素間の関係

分類	プロセス	目的	関連する取り組み
基盤	組織プロセス重視(OPF : Organizational Process Focus)	組織のプロセスおよびプロセス資産の現状の強みと弱みに対する綿密な理解に基づいて，組織のプロセス改善を計画し実施すること.	活動1 活動2
	組織プロセス定義(OPD : Organizational Process Definition)	利用できる組織プロセス資産の集合を確立し維持すること.	活動5
	組織トレーニング(OT : Organizational Training)	人員にスキルおよび知識を身につけさせることによって，人員が役割を効果的かつ効率的に遂行できるようにすること.	活動3 活動4
累進	組織プロセス実績(OPP : Organizational Process Performance)	「品質およびプロセス実績の目標」に応えるために，「組織の標準プロセスの集合」の実績に対する定量的な理解を確立し維持すること. また，組織のプロジェクトを定量的に管理するために，プロセス実績のデータ，ベースライン，およびモデルを提供すること.	活動6
	組織改革と展開(OID : Organizational Innovation and Deployment)	組織のプロセスおよび技術の測定可能な改善策として，漸進的な改善策および革新的な改善策を選択し展開すること. これらの改善策は，組織の事業目標から導出される組織の「品質およびプロセス実績の目標」を支援する.	活動2 活動6

次に，個々の構成要素で実現した内容を，CMMI の「プロセス管理のプロセス領域」の関係図に当てはめた.

図 5.3 は，基盤「プロセス管理のプロセス領域」に基づいて，活動 1～活動 5 の 5 つの活動の間の関係を整理したものである.

上級管理層からの目標を受け (a)，推進体制を確立するとともに，推進側のメンバのスキルを強化し (b)，全社的な SPI 活動の方針を落とし込む (c)．これらの取り組みは，主に活動 1 と活動 2 に関連した内容であり，OPF との関係が強い.

活動 4 と活動 5 に関連した取り組みは，OPD と強く関連する．SPI 活動推進のためのツールや手法を開発するとともに，全社で共有する資産の蓄積方法や展開方法を確立することが (d)，ここでの主な内容である.

SPI 活動推進のためのツールや手法に関するトレーニングコースを開発し，タイムリーに実施できる仕組みを構築する (e)．これは，主に，活動 3 に関連する取り組みであり，OT との関係が強い.

開発部門における SPI 活動のフィードバックが (f,g,h)，それぞれのプロセス領域に対して確実に入るようにしておくことが，個々の活動を改善するうえで重要である.

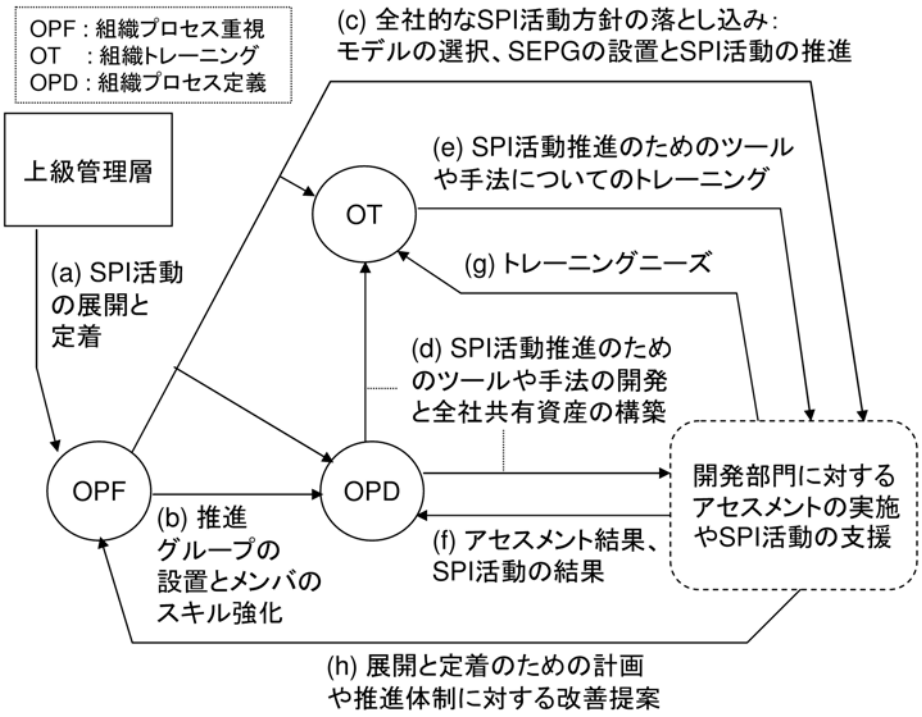


図 5.3: 基盤「プロセス管理のプロセス領域」に活動 1〜5 を当てはめた結果

図 5.4 は，活動 6 の仕組みと，全社的な SPI 活動推進のための体制や計画を見直すためのフィードバックサイクルを，累進「プロセス管理のプロセス領域」に当てはめたものである．

図 5.3 で示した取り組みをとおして，全社で共通的に測定する項目が定義される (i)．これが，OPP に対するインプットとなる．推進部門は，この測定項目の収集・分析とフィードバックの方法を定義したメトリクスプログラムとして提示し (j)，各開発部門における実施を支援する．メトリクスプログラムの実施結果として，各開発部門の SPI 活動状況とプロジェクト実績を収集し (k)，全社的な SPI 活動の推進に責任を持った管理者へ，進捗状況や達成状況を定期的に報告する (l)．また，各開発部門から提供されたデータは，全社の資産として蓄積される (m)．

図 5.3 で説明したことと同様，開発部門におけるメトリクスプログラムの実施結果が，それぞれのプロセス領域に対して確実に入るようにしておくことが (n,o)，全社的な SPI 活動推進に対する改善提案につながる (p)．この活動は，OID と強く関連する．

このように，提案した SPI フレームワークに基づいて実装した東芝版 SPI フレームワークの活動は，CMMI で提唱している考え方と一致していることが確認できた．

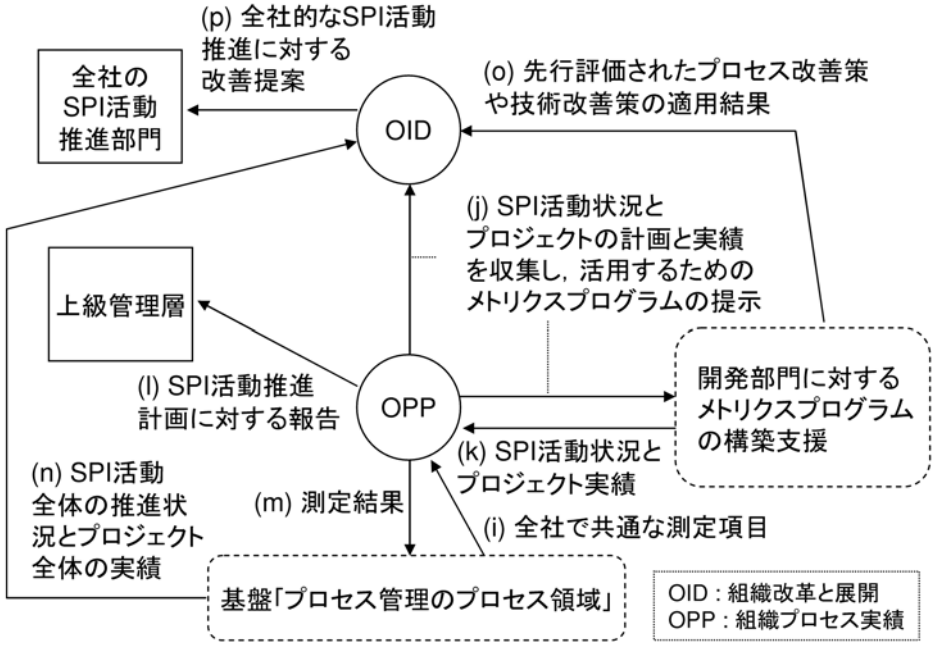


図 5.4: 累進「プロセス管理のプロセス領域」に活動 6 を当てはめた結果

5.7 東芝版 SPI フレームワークの実用性

全社レベルで SPI 活動を展開・定着させるために，表 5.1 で示した計画，実装，展開，定着のフェーズを，2000 年から 2010 年までの中長期計画として具体化し推進してきた．ここでは，3 章，4 章で示した活動状況や活動結果から，SPI フレームワークの実用性について考察する．

図 4.2 は，SEPG 設立部門数の推移を示したものである．2000 年度に活動を開始後，2008 年時点で，開発部門 SEPG の数は約 60 部門となっている．この数は，活動開始時点で SEPG の設置が必要と考えた部門数の約 80%に達している．また，カンパニー SEPG もほぼ組織化が完了し，3 階層 SEPG による体制が確立された．この結果から，全社レベルでの SPI 活動推進のための体制が構築され，維持されていることが分かる．

2005 年以降は，カンパニー SEPG が，コーポレート SEPG と協力して開発部門 SEPG を支援できるようになってきている．2010 年時点では，支援対象のうちの約 75%の開発部門に対してカンパニー SEPG が支援をしており[1]，開発部門に対する支援の中心が，コーポレート SEPG からカンパニー SEPG に推移している．表 4.2 で示したように，当初から定着フェーズではコーポレート SEPG のリソースを減らす計画をたてていたが，開発部門の自立化が進んだことと，カンパニー SEPG の整備が進んだことで，計画どおりのリ

[1]コーポレート SEPG と一緒に支援している部門を含む

ソースで推進できることを示せた．

CMM 関連のトレーニングは計画フェーズから提供し，その後，実施フェーズからは，SEPG リーダコースを提供してきた．図 4.4 に，CMM 関連教育，SEPG リーダコース，SQAG 関連教育の受講者数の累計を示す．CMM 関連教育の累計の受講者数は 1,000 人を超えている．また，SEPG リーダコースには，毎年，約 20 名の受講者が参加しており，累計の受講者数は 150 名を超える．SEPG リーダコースには，同一部門からの受講者も多く，SEPG リーダを担うことができる人材の育成が計画的に進められている．

開発部門内で SPI 活動を推進する際には，SEPG は改善活動を推進し，SQAG はプロジェクトや組織の開発プロセスのチェックや開発成果物のレビューなどを担うという体制を構築している．開発部門内で SPI 活動が継続的に行われ，開発プロセスが整備されてくると，SQAG に割り当てられる担当者も多くなり，SQA に関連したトレーニングの整備が求められてきた．そこで，2006 年から SQAG 入門コース，2008 年から SQAG リーダコースを提供し，SQA 活動を遂行できる人材の育成も進めている．

毎年開催するソフトウェアフォーラムは，社内外の状況を知って刺激を受け，新たな取り組みのヒントを発見する場として，SPI 関係者のみならず，ソフトウェア開発者に広く認知されている．2004 年以降は，コンスタントに 200 名以上の参加者があり，全社レベルのイベントとして認知されている．このフォーラムでは，社外からの講演と社内の事例発表が行われおり，情報共有だけでなく，各開発部門での SPI 活動の活性化に貢献している．

フォーラムにおける事例発表では幅広いテーマが扱われている．文献 [58, 守屋] の「定量的プロジェクト管理の実践」もその一つである．参加者が社内と限定されていることもあり，定量的プロジェクト管理に関する成功事例や失敗事例，ノウハウなどが具体的なレベルまで公開され，社内の定量的プロジェクト管理の展開に大きく寄与している．

図 4.18 は，2000 年からの社内公式アセスメントの結果の累計を年度毎に積み上げたものである．SPI 活動を推進している約半分の開発部門が，この社内公式アセスメントを利用している．これからも分かるとおり，レベル 3 以上の成熟度を達成した開発部門が年々増加している．このことから，開発部門における成熟度は着実に向上していると判断できる．また，社内公式アセスメントを利用していない部門でも，SPI 活動レポート作成のために収集するデータから，開発プロセスが着実に整備されているという実績が得られている．

また，SPI 活動が浸透し，SPI に関連する技術の習得が進むことによって，いろいろな改善活動の方法が提案されてきた．文献 [57, Iida] では，CMMI 連続モデルを活用し，課題ベース改善の中で診断を行い，その結果に基づいて改善項目を抽出するという方法が提案されている．また，文献 [48, Aoki] では，小規模の開発部門において，ワークショップをベースにした SPI 活動の具体的な推進方法と効果が述べられている．

ここでは，大規模組織において，SPI 活動を推進・定着させるための仕組みとして提案

した「SPI フレームワーク」に基づいて構築した東芝版 SPI フレームワークを利用して，約 10 年間にわたり SPI 活動を実践してきた結果を提示した．東芝版 SPI フレームワークを構成する個々の活動が，大規模組織において十分に認知され，機能していることを示した．また，全社的な SPI 活動の推進結果の評価として，成熟度の推移を示した．さらに，SPI 活動が浸透することよって，いくつかの改善方法が提案され実践されていることを示した．このような結果を総合的に判断すると，全社的に SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」とそれに基づいて構築した東芝版 SPI フレームワークは，大規模組織における SPI 活動の推進と定着に有効に機能していると判断できる．

5.8　10 年間にわたる全社的な SPI 活動の総括

SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」を提案し，そのフレームワークを活用して東芝版 SPI フレームワークを構築し，10 年にわたり実践してきた．なぜこのフレームワーク有効に機能したのかを，まず最初に考察する．その後，この仕組みの適用範囲について議論する．また，当初予定していて実現できていない活動の背景とその原因および解決案を検討する．

5.8.1　SPI 活動が定着した主な要因

開発部門で SPI 活動を推進する際には，SEPG の設置を必須とした．SEPG には，組織における改善活動の責任を持ち，改善技術／改善ノウハウの継続的な担い手となることを求めている．管理者には，組織の開発プロセスをよく理解していて，ソフトウェア開発に関する技術力があり，改善意欲の高いメンバを任命してもらうようにしている．成功要因のひとつとしては，各開発部門における SEPG の任命が適切に行われていたことが挙げられる．SEPG リーダコースの出席者に次期 SEPG リーダの参加が多かったことは，開発部門における SEPG リーダのローテーションが機能していることを示している．

ある活動を組織文化として定着させるにはかなりの時間が必要である．全社に向けて SPI 活動を推進する際に，2010 年までの中長期計画を，表 4.2，表 5.1 で示した 4 つのフェーズ（計画，実施，展開，定着）に分けて提示した．この中長期計画は，開発部門が SPI 活動を開始する際のコミットメントになっており，開発部門 SEPG に対し，一過性の活動ではないという安心感と信頼感を高めることに効果的であった．

表 4.2 で示した展開フェーズでは，カンパニー SEPG とコーポレート SEPG のメンバ数を，5.2.2 節で示した式に基づいて見積り，開発部門における SPI 活動の支援体制を特に強化した．これによって，開発部門に SPI 関連技術を短期間で，効果的に浸透できた．さらに，トレーニングコースやプロセス改善活動推進のための情報基盤を利用し，組織横

断的な人と人のネットワークを構築した．これは，SPI 活動に対する視野を広げるとともに，他部門のベストプラクティスの導入機会を増やすことに貢献した．

5.8.2 実現できなかった項目

本活動を進めるにあたり実現できなかった項目を以下に述べる．これらの項目は，今後の SPI 活動を推進するうえで検討すべき課題である．

1. アセスメント手法
   当初，CMMI に基づく診断を定期的に実施するために，アセスメント手法を構築し，開発担当者が他の開発部門の診断に参加できる仕組みの構築を目指した．しかし，開発担当者が他部門の開発プロセスを診断することのメリットを十分に説明できなかったため，この仕組みを確立することができなかった．

2. SPI 活動に対するスポンサー層の理解を深くしてもらうための活動
   SPI 活動に対するスポンサー層の理解を深めてもらい，コミットメントを強固なものとするために，イベントにおいてスポンサーセッションの実施を試みたが，スポンサー層に半日～1 日の時間を取っていただくことは難しく，数回実施しただけで終わってしまい，継続的に実施することができなかった．

3. カンパニー SEPG の構築と強化
   カンパニー SEPG は，3 階層 SEPG を構成する非常に重要な役割を果たす推進部門であるが，ビジネス環境の変化やカンパニーの人員計画などの影響によって，十分なリソースと資金が提供できない場合も少なくない．SPI 活動の必要性と効果を経営層により良く理解していただくとともに，コーポレート SEPG とのローテーション強化などをはかり，カンパニー SEPG の位置付けを強固なものにする必要がある．

4. SPI 関連のスキル定義とキャリアパス
   全社的な SPI 活動を推進してきたことで，SPI 活動に関与するメンバが増えてきた．それにともない，組織の中で，SEPG や SQAG の役割は認知されてきたが，メンバのスキル定義やキャリアパスなどは十分に示せていない．

5. 経営層からの継続的なコミットメント
   本活動の立ち上げ時は，経営層のコミットメントも高く，見積りと同等の予算も配分されたため，さまざまな活動を同時並行で加速することができた．約 10 年間の活動の中間時点で，コーポレート SEPG のスポンサーが交代となった．新しいスポンサーは，全社的に SPI 活動を推進・展開することに理解は十分に示してくれていたが，活動の成果が果たして上がっているのかどうか，推進メンバのモチベーション

が維持されスキル向上がされているのか，という点を懸念していた．客観的に自分たちの活動を振り返ってみると，自分たちの価値やスキル，達成した成果についての考え方に自分本位のところがあった．コーポレート SEPG のスポンサーからのコミットメントが得られなくなったというわけではないが，コミットメントを得るための説明や活動成果の提示が弱かったのは事実である．

5.8.3 今後の課題

成熟度の高い開発部門では，SPI 活動の成果と効果を定量的に示せるようになってきた．ただし，ソフトウェアの開発には多種多様な要因が絡むため，品質・納期・コストの予実差や，生産性や欠陥密度を単純に計算しただけでは有意な差が得られないこともある．全社レベルでの SPI 活動に対するコミットメントを強固にするために，SPI 活動の実績と開発プロジェクトのデータを関連付けた分析方法の確立が，今後の大きな課題である．

また，開発部門の成熟度が高くなると，推進部門側に対する要望もソフトウェア生産技術や定量的管理技術の導入などを含む高度なものになってきた．そのため，推進部門側としては，ソフトウェア生産技術の知識と経験を深めるとともに，ソフトウェア生産技術の研究や開発を担っている部門との連携も含め，求められる技術をタイムリーに提供できる体制の強化が必要である．

第6章　結論

本章では，本研究の課題に対して設定した目的の実現度合い，実践した内容の妥当性をまとめる．さらに，今後の進むべき方向性を述べる．

6.1　目的の達成度

本研究の課題と目的を以下に再掲する（詳細は1章を参照のこと）．

大規模組織においてSPI活動を展開・推進するものの，期待された効果が得られない原因は，以下のように考えられる．

- **原因**

  ソフトウェア開発のベストプラクティスであるCMMIが広く普及しているが，それを利用して組織横断的にSPI活動を推進するための方法論が確立されていない．

このような状況の中，企業においては，組織横断的にSPI活動を確実に，そして効果的・効率的に実現する方法が求められている．

そこで，本研究では，上述したSPI活動の推進を阻害する原因の解決を目的とし，以下の項目に関する研究を行った．

- **目的**

  企業内において組織横断的にSPI活動を推進・展開するための方法論の確立

この目的を達成するために，SPI活動を推進・定着させるための仕組みである「SPIフレームワーク」を提案した．このSPIフレームワークは，1990年代にいくつかの品質管理手法／ツールを社内展開した際，手法やツールの利用が組織横断的に定着した組織の特徴を分析して導出したものである．このフレームワークは，6つの要素から構成されている．

提案したSPIフレームワークに基づいて，東芝版SPIフレームワークを構築し，10年間にわたり全社的なSPI活動を実践してきた．その結果は，3章〜5章に整理した．3章では，SPIフレームワークの提案とそれに基づいた具体的な実装結果を示した．4章では，提案したSPIフレームワークに基づいて構築した東芝版SPIフレームワークを活用した全社的なSPI活動の実践結果を示した．さらに，5章では，10年間にわたり実践した結果から得られた内容を考察した．

これらの結果からも分かるとおり，提案した SPI フレームワークおよびそれに基づいて実装した東芝版 SPI フレームワークとも汎用性は高く，筆者が所属する組織の独自性による影響は大きくないと考えている．つまり，提案した内容および実装・実践結果は他の組織でも十分に活用できるものである．

また，東芝版 SPI フレームワークを構築して推進してきた全社的な SPI 活動の推進の結果，活動当初に設定した目的が達成できたことを確認した．つまり，東芝版 SPI フレームワークの実用性が確認できたといえる．

これらの評価結果を総合すると，“組織横断的に SPI 活動を推進するための方法論が確立されていない”という課題に対して，本研究で提案した SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」がひとつの解決策であると判断できる．

6.2 適用範囲

SPI 活動を開始する際の一番のポイントと考えたことは，いかに多くの開発部門にこの活動を展開し定着させることができるのか，ということを 3 章で述べた．SPI 活動を展開し定着させるために構築した東芝版 SPI フレームワークは，大規模組織においては有効に機能することが実証できた．文献 [56, Sugahara] では，企業の枠を越えて開発プロセスに関する研究や議論をする場の重要性が示されている．大規模組織では，個々の組織文化や製品ドメインは大きく異なるため，企業の枠を越えた活動と類似点は多い．企業の枠を越えた体制や仕組みを構築する際には，今回提示した仕組みが適用できるはずである．

また，SPI 活動を推進する際には，SPI フレームワークで提案した 6 つの要素に基づく活動は欠かせないものである．本論文で示した SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」を参考に，類似の仕組みを構築し，運用しているカンパニーもある．これは，中小規模の組織においても，SPI 活動を推進するために，本論文で提案した「SPI フレームワーク」が利用できるというひとつの証拠でもある．また，このフレームワークの汎用性が高いことは，5.7 節でも示している．

6.3 将来性

開発プロセスが安定しない，すなわち，手順として実施すべきことが決まっていても，外乱によってその手順が安易に変わってしまうような状態で，先端のソフトウェア生産技術や定量的な管理技術などを組織的に導入し，定着させることは難しい．

3 章では，筆者らが 1990 年代にソフトウェア品質管理のための手法やツールの普及・展開のための活動を推進していたことを述べた．開発部門 SEPG とカンパニー SEPG の設置が落ち着いてきた 2005 年以降，静的解析ツール [61] や構成管理ツール [50]，テスト管

理ツール [86][87]，レビュー技術を導入する開発部門が増え，多くの開発部門でこれらの技術が定着している．この結果に対しては，カンパニー SEPG や開発部門 SEPG の存在およびその役割が大きく貢献している．

技術の進展が早いソフトウェアの開発現場においてこそ，SPI 活動をしっかりと根付かせ，改善活動を常態化させることが重要であると考えている．SPI 活動推進のための仕組みをさらに発展させることが，ソフトウェア生産技術や定量的な管理技術の効果的・効率的な推進と定着に寄与するはずである．

6.4 今後の進むべき方向性

本研究では，多くのソフトウェア開発プロジェクトで問題が発生し，さらにその問題解決のための改善活動がなかなか展開・定着しないという現状を踏まえ，まず，ソフトウェアプロセス改善の基本的な考え方と問題点を整理し，その解決のために強化して取り組むべき活動を示した．

本研究で提案したソフトウェアプロセス改善活動を推進・定着させるための仕組みである「SPI フレームワーク」は，既存のいろいろなアイデアや実践事例の中から抽出し組み合わせたものである．提案した SPI フレームワークに基づいて実装した東芝版 SPI フレームワークを構成する一つ一つの要素を見ると，新規性の高いものばかりではないが，それらを有機的に結びつけてひとつの形として仕上げたことに価値があると考えている．さらに，東芝版 SPI フレームワークを活用し，企業における SPI 活動に約 10 年間という長期にわたる期間に適用し，その実用性と有効性を示したことに意義がある．

今後の課題として以下の研究を進めていく必要がある．

ソフトウェアのプロセス改善は成功する組織がある一方，あまり効果を上げられていない組織も多い．成功する組織はたまたま偶然に成功するのではなく，それなりの工夫や知識の蓄積があって成功している．しかしそれらの知識は大半が経験知，暗黙知の領域にとどまっており，多くの組織がその知識を共有するに到っていない．そこで次の二つの観点から，経験知，暗黙知を明示知としてまとめ，多くの組織で知識の共有化ができるようにしていかなければならない．

第一の観点としては，活動の成果と効果をいかに示すのかということである．成熟度の高い開発部門では，SPI 活動の成果と効果を定量的に示せるようになってきた．ただし，ソフトウェアの開発には多種多様な要因が絡むため，品質・納期・コストの予実差や，生産性や欠陥密度を単純に計算しただけでは有意な差が得られないこともある．全社レベルでの SPI 活動に対するコミットメントを強固にするために，SPI 活動の実績と開発プロジェクトのデータを関連付けた分析方法の確立が，今後の大きな課題である．

第二の観点としては，経験知，暗黙知を明示知として取りまとめ，整理して文書化するなどをして形に残す役割の明確化とその役割を遂行するために必要となる組織体制の構築である．東芝版 SPI フレームワークで構築した「3 階層 SEPG」は，この役割を担う能力を持っていると確信している．SPI 活動を推進する際には，適切な階層で SEPG の役割を担う体制の構築が重要なポイントである．また，SPI 活動が定着してくると，開発部門からの要望も，開発プロセスの改善や構築だけではなく，ソフトウェア生産技術や定量的管理技術の導入などを含む高度なものになってくる．そのため，推進部門側としては，ソフトウェア生産技術の知識と経験を深めるとともに，ソフトウェア生産技術の研究や開発を担っている部門との連携も含め，求められる技術をタイムリーに提供できる体制の強化が必要である．

1990 年代後半から 2000 年代前半までは，CMM がかなり注目を集め，多くの企業で導入されてきた．本論文で詳述した活動の実施に際しては，CMM が非常に大きな役割を果たしている．そして，本論文で示したとおり，この 10 年間の活動では，一定の成果が得られた．現在，プロセス改善の領域においては，CMM のような影響力のある技術は存在しない．今後求められることは，次の 10 年の計画を早急に立案することである．これは，今までプロセス改善活動の普及・展開に関わってきた人々に対する大きなチャレンジでもある．今までの 10 年間の活動をしっかりと振り返り，蓄積してきた経験を生かした計画の立案が求められている．

本論文では，大規模組織において，開発部門における SPI 活動の展開と定着を促進するために，SPI 活動を推進・定着させるための仕組みである「SPI フレームワーク」を提案し，そのフレームワークに基づいて構築した「東芝版 SPI フレームワーク」を使って SPI 活動を実践してきた内容を提示した．さらに，10 年間にわたる実践結果から，提案した SPI フレームワークの汎用性と東芝版 SPI フレームワークの実用性とを実証した．

ここに紹介し実践した内容が，この種の活動を展開しようとする企業や組織の参考になれば幸いである．

付録

付録には，東芝版 SPI フレームワークにおける改善ソリューションとして提供してきた 2 つの手法に関する研究・開発の結果を示す．この結果は，図 3.12 に示したステップ 1「手法／ツールの分析や開発」の活動に対応するものである．

付録 A では，メトリクスを活用した品質管理手法を提案し，その手法を実際の開発に適用して効果を示した．さまざまなメトリクスが提案されているが，シンプルなメトリクスを継続的に計測し，そのメトリクスを活用するための運用方法を明確にすることで，品質の観点から非常に効果のある活動につながることを提示した．また，ツールとそのツールを活用した運用プロセスを構築することが，組織内での継続的な利用においてはポイントになることを示している．

付録 B では，プログラム静的解析技術の適用事例である．プログラム静的解析ツールの解析能力は非常に高いものがあるが，ソースコードの規模と比較してかなり膨大な警告メッセージが検出されてしまうという問題があった．この問題を解決するために警告メッセージを絞込み，開発者が容易に利用するためのシステムを構築し，多くの開発部門に展開できたことを示した．

付 録A　メトリクスを活用した品質管理手法

ソフトウェアの品質メトリクスは，数多くの提案がなされている．メトリクスの効果は，成果物やプロセスを客観的に，そして正確に把握することができることである．実際の開発現場でも，メトリクスを利用したソフトウェアの管理がかなり浸透している．しかし，メトリクスを計測することの負荷や個人差が含まれているという問題は，完全に解決されているわけではない．

そこで，無理なく，簡単に，しかも自動的に計測できる品質メトリクスを利用した，ソフトウェアの品質管理を提案している．これは，ソフトウェア成果物の品質を定期的に計測し，レビューを行うことによって，各工程で品質を作り込むことを目的としている．

ここでは，品質メトリクスを利用して，ソフトウェアの設計工程からテスト工程までを，モニタリングし，品質の作り込みに利用するためのモデルと適用事例を紹介する．

適用の結果，品質メトリクスの活用により，各工程の作業や成果物の問題を早期に発見，除去できることが明らかになった．また，定量的データを用いた定期的なレビューの実施により，プログラミングの構造に良い影響を与えることが明らかになった．さらに，テスト工程では，不具合情報と品質メトリクスを併用することによって，テストの状況を素早く，正しく把握できることを明らかにした．

A.1　はじめに

一般に，ソフトウェアの品質は，テスト工程における不具合，テスト項目数，テストカバレッジを中心に評価されている．しかし，テスト工程を中心とした品質の作り込みでは，かなりの後戻り工数が必要になる場合が多い．大規模化，複雑化しているソフトウェアを考えた場合，ソフトウェアの品質は，上流工程から確実に作り込む必要がある．また，テストも，システムの特徴に合わせて工夫しなければならない [70][71][31].

無理なく，簡単に，しかも自動的に計測できる品質メトリクスを利用した，ソフトウェアの品質管理方法を提案してきた．その提案にもとづいて開発したツールがESQUT(Evaluation of Software Quality from User's viewpoinT) である．これは，図的表現で記述した詳細設計書およびC，COBOL のソースコードを計測するツールである．開発成果物を自動的に計測できるため，利用者は無理なく，容易にメトリクスを計測することができる．

ここでは，ESQUT を利用した品質管理モデルを提案する．そして，実際のプロジェクトへ適用した結果と評価について説明する．

A.2 ソフトウェアメトリクスの提案

ソースコードの品質メトリクスとして，McCabe[67] や Halstead[68] の提案したメトリクスが有名である．McCabe のメトリクスは，制御の流れのグラフ表現におけるパスに注目した尺度である．Halstead のメトリクスは，オペレータとオペランドの種類に注目した尺度である．しかし，これらのメトリクスは，開発者自身が計測結果を見て，レビューに利用するには，内容を直感的に理解できない面があり，開発担当者が利用するには問題が多い．メトリクスは，簡単に計測でき，かつ利用容易でなければ，実際の開発現場での利用は期待できない．開発者が理解しやすく，計測が容易な品質メトリクスを，以下の観点から導き出している [66].

- メトリクス抽出の観点
    1. モジュールの複雑さ
    2. モジュールの機能性
    3. モジュールの理解容易性

上記の観点から設定した C ソースコードのメトリクスを以下に示す.

- ファイルレベル
    1. モジュール数
    2. 総ステップ数
    3. インクルードファイル数
- モジュールレベル
    1. ステップ数
    2. 条件分岐数
    3. ループ数
    4. 引数数
    5. コメント行数

これらのメトリクスと，McCabe や Halstead のメトリクスとの相関が高いことは，論文 [65] で報告されている．また，ソースコードだけではなく，グラフィカルな表現で，プログラムのアルゴリズムと手順を表した詳細設計書の品質を計測できる．このグラフィカルな表現の作成を支援するツールとして，TFF-Tools を開発済みである．TFF-Tools(TFF:Technical

description Formula for Fifty steps / module design) で記述された設計書の品質計測結果と，C ソースコードの品質計測結果の相関が高いことも，論文 [65] で報告されている．詳細設計書の品質メトリクスを以下に示す.

- システムレベル
    1. モジュール数
    2. 総プロシージャ数
- モジュールレベル
    1. プロシージャ数
    2. 条件分岐数
    3. ループ数
    4. 引数数

A.2.1 プログラム品質評価ツール ESQUT の概要

ソフトウェア品質評価ツール ESQUT は，A.2 で示した品質メトリクスを計測するツールである．このツールの特徴を以下に示す.

1. プログラムの自動計測
2. ソフトウェアの品質定量化
3. 定期的計測結果に基づく管理情報の提供
    (a) 進捗情報
    (b) 比較情報
    (c) 基準値オーバー情報
4. テスト品質評価
5. 品質帳票の作成

このツールの概要を，図 A.1 に示す．計測機能では，計測日指定ファイルで設定した日付に合わせて，対象成果物を自動的に計測する．これらの情報は，定期的に定められた場所に蓄積される．この情報を利用して，品質帳票を作成する．

また，定期的に計測された複数の計測結果から，進捗状況，比較情報，基準値オーバー情報を抽出する．進捗情報では，プログラムのファイル数やステップ数の推移をビジュアル化する．比較情報は，複数回の計測結果の相違を一覧表示する．さらに，基準値オーバー情報では，ユーザが設定した品質基準値を超えている計測結果を一覧表示する．

テスト品質評価機能では，テストの消化状況と不具合検出状況と，プログラム構造の変化（ステップ数や条件文数の変化量）などを合わせて表示する機能を提供している．

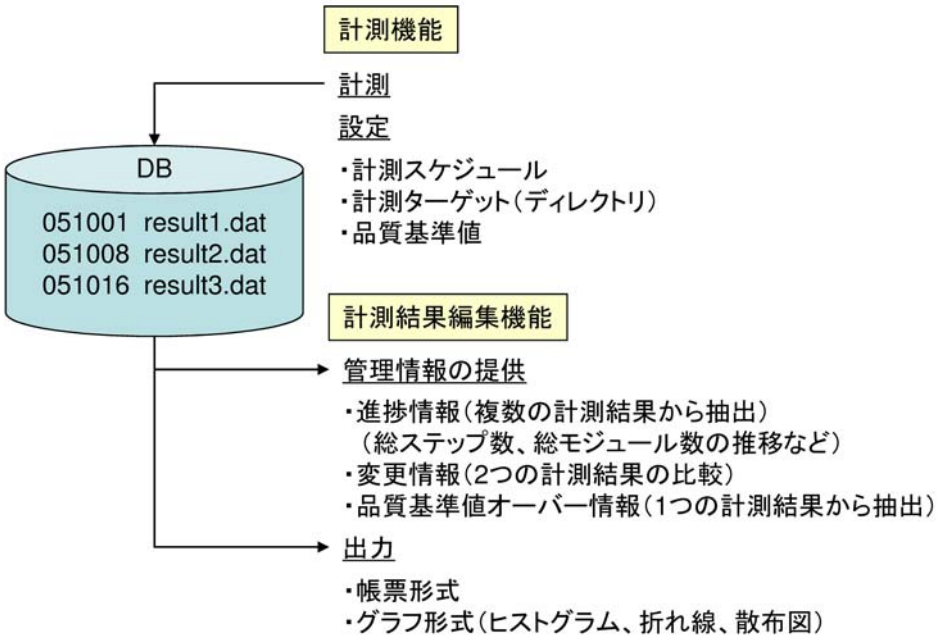


図 A.1: 品質評価ツール ESQUT の機能概要

A.3　メトリクスを利用した定量的な品質管理

ここで提案するモデルの基本は，すべての工程で定期的にPDCA(Plan-Do-Check-Action)の品質管理サイクルを回して，品質の作り込みを行うことである．

Plan の段階で設定する品質基準値は，過去のデータや経験にもとづいて設定する．品質基準値は，計測メトリクスごとに設定するものと，メトリクスの組合せにより設定するものがある．メトリクスの組合せによる品質基準値の例を，以下に示す．

- コメント率：コメント行数／ステップ数
- 条件文数率：条件文数／ステップ数

品質記述値を超える計測データの品質に問題が含まれている場合が多いことは，A.5.1 で説明する．

A.4　定量的な品質管理モデル

定量的な品質管理を行うために，ESQUT を利用して，ソフトウェアの設計工程からテスト工程までをモニタリングし，品質作り込みに適用する．

このモデルの目的は以下のとおりである．

1. ソフトウェア生産物の定量化
   各工程の成果物の品質を定量化し，さまざまな視点からのデータを提供する．
2. 定期計測による品質のモニタリング
   計測結果から，問題の早期発見を行い，レビューにより問題の解決を行う．
3. 設計工程からテスト工程までの品質評価
   各工程ごとに品質を作り込み，後工程に問題を残さない．

提案する定量的品質管理モデルを，図 A.2 に示す．基本的なメトリクスと，それらの時間的な推移に基づくメトリクスを持つ．それぞれのメトリクスに対応した活動を，各工程ごとに実施する．

詳細設計書やプログラムの品質メトリクス計測データの分布を調べ，品質基準値をオーバーしているプログラムを検出する．検出した詳細設計書やプログラムは，レビューを実施し，内容を確認する．特に，サイズの大きなモジュールは，機能分割ができていない可能性が高いので，必ずレビューを実施する．

A.4.1 設計／コーディング工程における活動

設計，コーディング工程では，変化量の大きなモジュール，変化量が連続して発生しているモジュールに注意する．逆に，変化が小さい場合は，作成作業やレビューが適切に行われているかどうかを確認する．

A.4.2 テスト工程における活動

1. テスト開始時点
   作成工程の最終段階で，計測したメトリクスの結果から，プログラムの複雑さを評価する．その評価結果に基づき，テスト工程で検出される不具合数を予測する．さらに，テスト項目数の十分性を確認する．
2. テスト期間中
   定期的にメトリクスを計測し，プログラムの構造の変化をチェックする．この情報と不具合情報（検出数，修正数）から，テスト状況を把握し，テストの品質を管理する．時間的な変化量が大きい場合には，レビューを行い，テストケースの追加を検討する．逆に，変化量が小さい場合には，テストの内容が適切かどうかをレビューして確認する．テストが十分で，変化量が連続して小さくなれば，プログラムが安定してきたと判断できる．
3. テスト終了時点
   テスト開始前の予測と実績を比較して，量的基準を満たしているかどうかを確認す

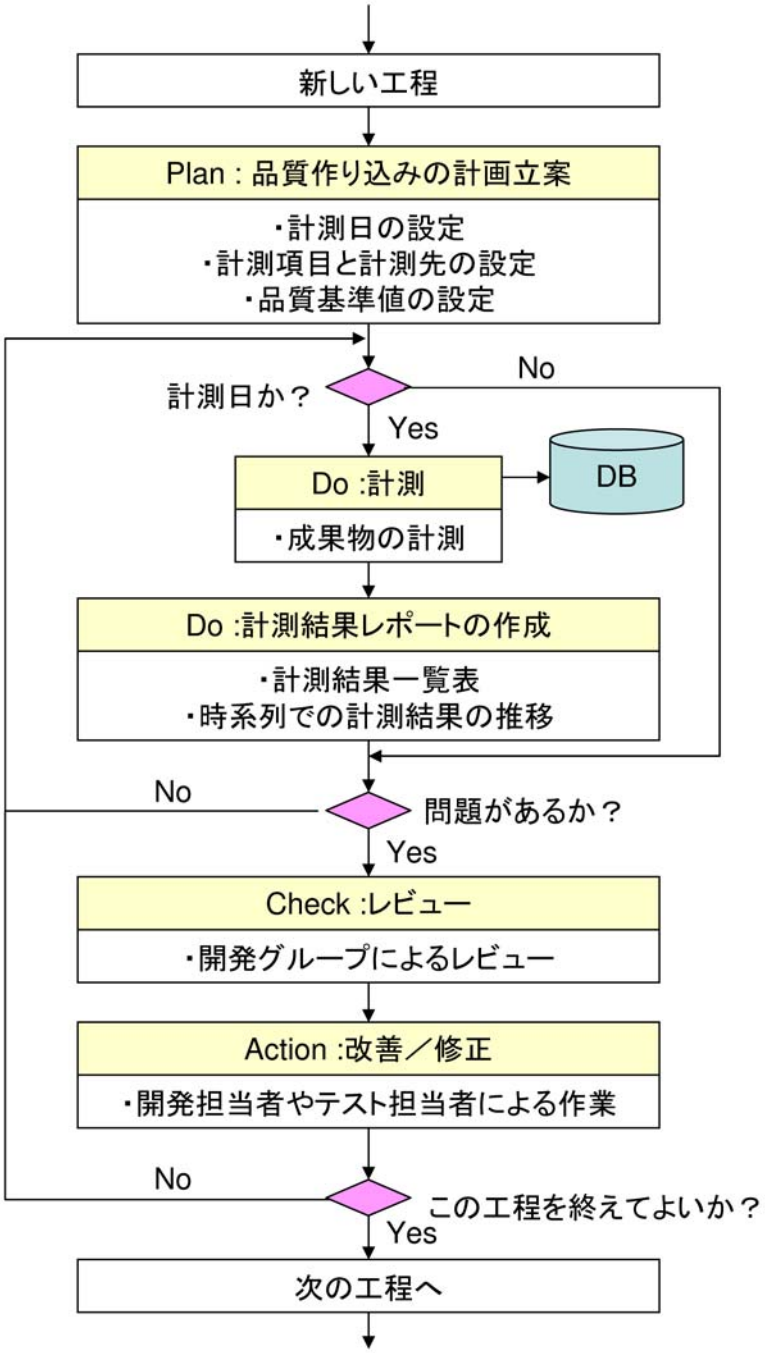


図 A.2: 定量的品質管理モデル

る．この結果は，データベース化し，予測の精度向上のために利用する．

A.5　適用事例

ここでは，ESQUT を実プロジェクトに適用した結果を示す．プロジェクト毎の適用の目的を，以下に示す．

- プロジェクト 1（A.5.1 参照)
  品質メトリクスの効果を確認するため，設定した基準値を超える計測データが，品質に与える影響を確認する．
- プロジェクト 2（A.5.2 参照）
  定期的に品質メトリクスを計測し，レビューに利用することによる効果を確認する．
- プロジェクト 3（A.5.3 参照）
  テスト工程での品質メトリクスの利用方法を検討し，効果を確認する．

A.5.1　品質メトリクスの有効性

設計工程の品質が悪ければ，後工程のテスト工程の品質も悪いことが予想される．ここでは，設計工程の品質メトリクスで基準値を超えているものが，テスト工程で，どれだけの不具合を出しているのか確認する．これによって，品質メトリクスの有効性を明確にする．

1. 対象製品の概要
   - 製品　：通信機器
   - 設計　：TFF 利用
   - サイズ：32Kbyte(ROM size)
   - 分類　：新製品（シリーズ品）
2. 開発済みシステムの分析
   設計，プログラムの方法に大きな変化がなければ，シリーズ製品の場合，プログラムのサイズ，構造には大きな変化がないと考えられる．そこで，過去に開発したシリーズ製品で，規模が同程度のものから，品質基準値を導出した．
   過去に開発したシリーズ製品について，ESQUT メトリクスを計測した．各メトリクスの関係を，図 A.3 に示す．
   各メトリクスの基本統計量を，表 A.1 に示す．
3. 品質基準値の設定
   品質基準値を低い値に設定すると，問題がありそうなモジュールとして指摘される数が多くなる．問題を絞り込んで，特に作りが複雑と考えられるモジュールだけを抽出するために，以下の方針にもとづいて，モジュールの品質基準値を設定した．

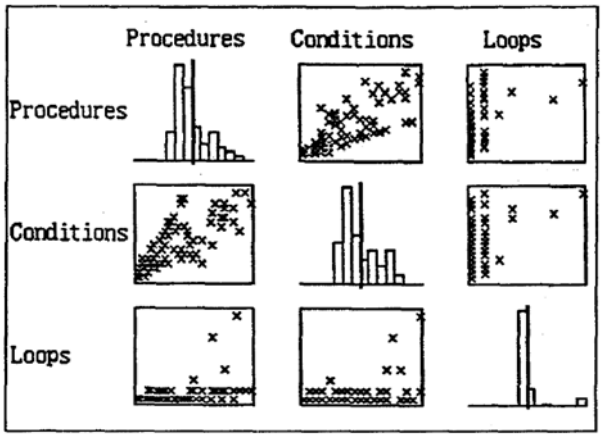


図 A.3: 各メトリクス間の関係

表 A.1: 各メトリクスの基本統計量

メトリクス	最小	最大	平均	標準偏差
処理 BOX 数	2	42	3.9	10.21
条件 BOX 数	0	18	6.2	5.15
ループ数	0	8	0.7	1.65

- 標準偏差を利用する.
  設定例 :「品質基準値＝標準偏差± 2 σ」
- 累積度数 90%を超える値. 図 A.3 の処理 BOX 数 (プロシージャ数) と条件 BOX 数の場合.
- 平均から大きくはずれている値. 図 A.3 のループ数の場合.

上記の方針にしたがって設定した品質基準値を，表 A.2 に示す.

表 A.2: 品質基準値

メトリクス	品質基準値
処理 BOX 数	30
条件 BOX 数	14
ループ数	4

4. 品質メトリクスの利用方法
   個人差によるバラツキを少なくし，ソフトウェアの品質を均一化させるために，定期的に計測し，基準値内での設計を推奨した. 計測結果にもとづくレビューは，必要に応じて実施した.
5. 品質メトリクスの効果

品質基準値を超えた計測データに含まれている不具合数を調べるために，システムテスト工程で検出された不具合の内容を分析した．
図 A.4 に，総モジュール数に占める基準値オーバーモジュールの割合を示す．

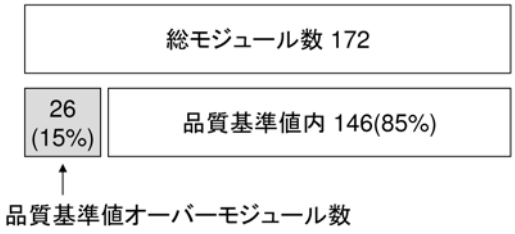


図 A.4: 総モジュールに占める基準値オーバーモジュールの比率

図 A.5 に，総不具合数に占める基準値オーバーモジュールの割合を示す．

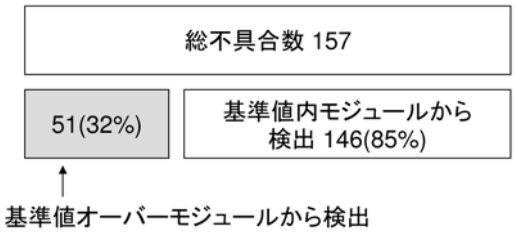


図 A.5: 総不具合数に占める基準値オーバーモジュールの比率

基準値を超えたモジュールは 26 件（15%）である．そのモジュールから，総不具合数 157 件のうち 51 件（32%）の不具合が検出されている．これは，定期的にメトリクスを計測し，基準値を超えたモジュールをレビューすることによって，かなりの効果が期待できることを示している．

A.5.2　品質モニタリングの有効性

作成工程のメトリクスを定期的に計測し，計測結果にともついてレビューを実施した．定期的なメトリクスの計測と，その結果を利用したレビューの効果を確認する．

1. 対象製品の概要
   - 製品　：CAD
   - 言語　：C 言語
   - サイズ：10Ksteps（機能の一部）
   - 分類　：追加機能（流用を含む）

2. 品質基準値の設定

過去に開発したシリーズ製品のメトリクスを計測し，品質基準値を設定した．設定した品質基準値を，表 A.3 に示す．この製品は，CAD の処理を行うものであり，複雑な処理を必要とする．

表 A.3: 設定した品質基準値

メトリクス	品質基準値
ステップ数	170
条件文数	30
ループ数	10
引数数	6

3. メトリクスの利用方法

メトリクスを定期的に計測し，計測結果に基づいて定期的にレビューを実施した，各レビューごとに，基準値オーバー，進捗，新規，変更，および削除の各モジュール情報を表とグラフで示した帳票を作成した．レビューでは，リーダが各開発担当者のデータをチェックし，現状の進捗状況を確認した．また，複雑なモジュールへの修正および，変更量の大きなモジュールは，特に注意して確認した．保守性という観点から，コメント率にも注意を払った．コードレビュー（内容のチェック）は，必要に応じて行った．

各レビューごとに用意した品質帳票の例を，図 A.6 に示す．この帳票には，進捗，新規，変更，および削除の各モジュールの内容が示されている．

計測は，開発工程の途中から行った．10 日に一回の間隔で計測し，定期的にレビューを実施した．

4. 定期レビュー実施による効果

定期的な計測と，その計測結果にもとづくレビューの実施によって，プログラム構造の変化が明らかになった．プログラムを定量化して，比較評価することで，作り方に対する意識の変化が現れたことによるものだと考えられる．新規に作成したモジュールと，開発済みのモジュールのプログラム構造の比較結果を，図 A.7 に示す．システム全体では，新規に作成したモジュールで基準値を超えたものはなかった．基準値を設定することによって，モジュール化に対する意識が働いたことも，基準値オーバーモジュールがでなかった一つの要因と考えられる．

ソフトウェアの成果物を定量化し，レビューで利用することによって，早期の段階から品質の作り込みが可能になる．また，プログラムが複雑になって，理解が困難になる前に，指摘して改善することが可能となるため，品質の向上が期待できる．

Product Name	ESQUT
Check Date	2005.09.25

	File Name	Date
1st	esqut-1.rpc	2005.09.15
2nd	esqut-2.rpc	2005.09.22

Directory Name	c:¥home¥source¥ESQUT

[New Modules]

File name	Module name	Step	Cond	Loop	Arg	Comment
dtyp_chk.c	dtyp_chk_sub	101	25	9	0	32
	dtyp_chk_mcr	62	13	7	0	20
	dtyp_err_set	21	1	0	2	7
rck_tbl.c	teigi_dat_set	49	8	1	8	26
	aut_get_call	26	5	0	6	25

[Changed Modules]

File name	Module name	Step	Cond	Loop	Arg	Comment
qsnddata.c	asnddata	89	9	0	6	58
		94	8	0	6	69
snddata.c	snddata	69	8	0	6	19
		73	8	0	6	23
xcongens.c	xcpmgems	43	12	0	1	6
		56	21	0	1	6

図 A.6: 品質帳票の例

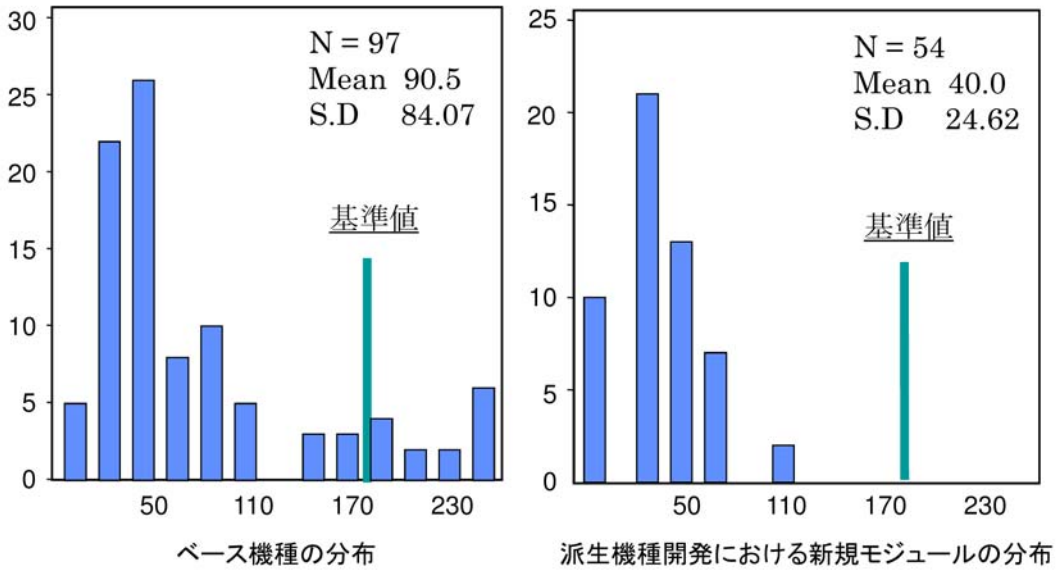


図 A.7: プログラム構造の変化

A.5.3 テスト工程における利用方法と効果

ここでは，テスト工程に ESQUT を適用して，テスト工程の品質管理に利用する方法と，効果について分析した結果を示す.

1. 対象製品の概要

   - 製品　：通信機器
   - 言語　：C 言語
   - サイズ：700Ksteps
   - 機能数：13

2. 品質メトリクスの利用方法

   テスト工程では，一週間に一回の割合で計測を行った．定期的に計測結果から，ソースプログラムの変化量の時間的推移の情報が得られる．この「ソースプログラムの変化量」と「不具合情報」を併用して，プログラムの品質を評価し，利用方法を検討した．計測結果にもとづくチェックの観点を以下に示す.

   - 不具合情報と変化量のズレが大きい機能
   - 条件分岐が増減した機能
   - 変化量が大きな機能

3. プログラム構造の変化

   テストの進捗にともなって，プログラムの構造が変化した結果を図 A.8 に示す．この図は，プログラムに含まれる総ステップ数と総条件文数の推移を示している．総ステップ数の変化量と一緒に，条件文数やループ数の推移を確認することで，プログラムの変化内容が把握できる．図 A.8 の場合は，ステップ数の変化量と条件文数の変化量は，ほぼ同じ傾向を示している．5 週目 (Week-5) と 10 週目 (Week-10) の期間で，かなり大きな変更が入っている．これは，機能の追加や修正が多いことを示している．

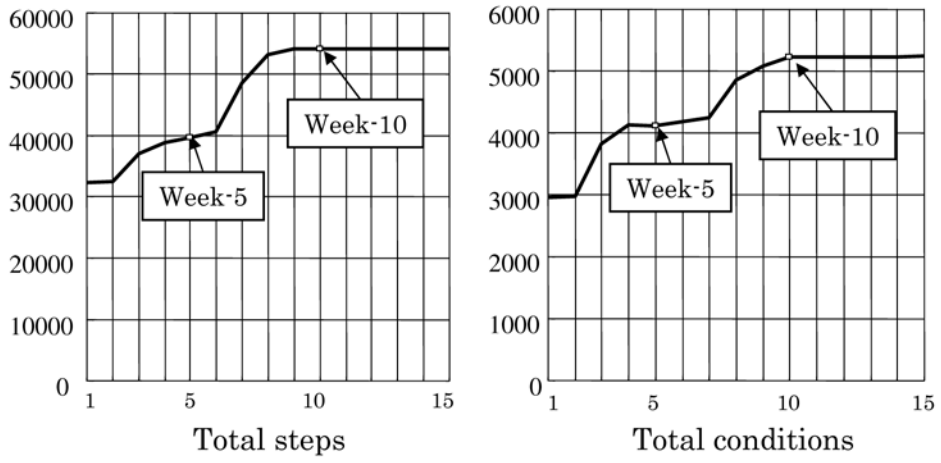


図 A.8: メトリクスの推移

4. プログラム構造の変化と不具合数

   プログラム構造の変化にともなって，不具合がどのように変化しているのか確認した．累積不具合数の推移を，図 A.9 に示す．ここでは，9 週目 (Week-9) と 16 週目 (Week-16) の時点で集計した結果を示している．この図の左側のグラフが示すとおり，5 週目 (Week-5) と 10 週目 (Week-10) の間は，変化量も多く，多くの不具合が出るはずである．9 週目 (Week-9) に作成した累積不具合数では，作成時点のプログラム品質が安定した状態であることを示している．

   しかし，16 週目 (Week-16) の時点で累積不具合数を見ると，30 件以上の不具合数が，9 週目 (Week-9) の時点で報告されていなかったことが分かる．

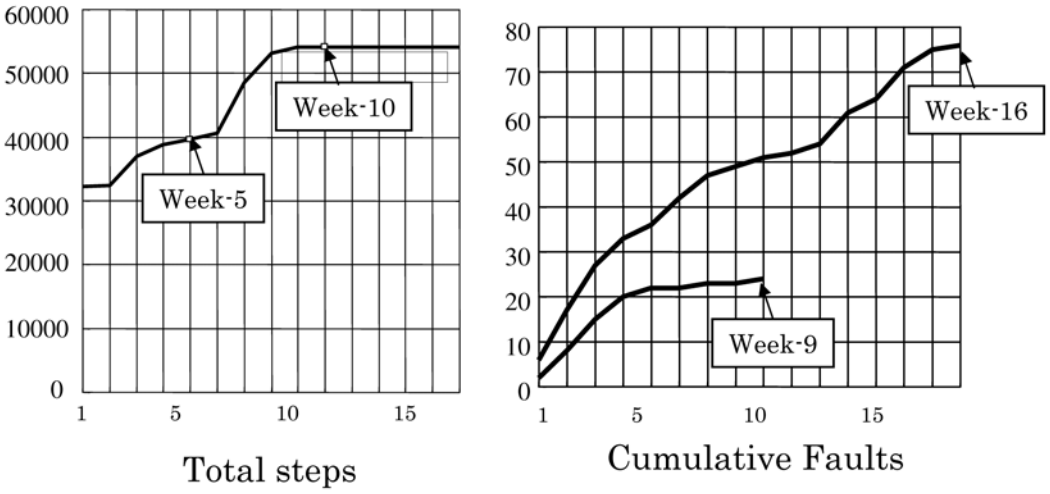


図 A.9: 累積不具合数の推移

   このような現象は，以下に示す手順で不具合報告が処理されるため，タイムラグが生じていることを示している．

   - 不具合を発見
   - 不具合管理報告書を作成
   - 管理シートに登録

5. メトリクス利用による効果

   テスト工程で，メトリクスを利用することの効果について考察した結果を，以下に示す．

   - リアルタイムな情報

     図 A.9 からも分かるように，不具合情報を中心としたテスト工程の管理では，リアルタイムな管理ができない．また，不具合情報には，質と量に個人差が含

まれる場合が多く，精度がよいとはいえない．不具合情報とプログラム構造の変化量を併用することで，リアルタイムに，しかも精度よく，客観的な管理が実現できる．

- テスト工程の安定化傾向の把握

  図 A.8 から分かるように，ある時点から，プログラムの変化量はなくなっている．これは，テスト工程が安定してきたことを意味する．プログラムの品質が安定してきたかどうかは，不具合情報と併用することによって，確認することができる．

- 不具合原因の種類と影響の大きさの把握

  条件文数やループ数の推移により，プログラム構造を確認することによって，不具合の原因を推定できる．

  プログラム構造の変化から，不具合原因の種類を推定する方法を，表 A.4 に示す．

表 A.4: 不具合原因の種類の推定方法

No.	項目	不具合の原因
1	規模の増加	仕様の抜け 仕様／設計の誤り
2	条件分岐の増加	判定すべき条件の抜け エラー処理の誤り ロジックの変更
3	ループの増加	配列等の初期化などの抜け ロジックの変更
4	引数の増加	モジュールの再分割 モジュールインタフェースの変更 モジュール設計誤り

A.6 まとめ

ESQUT を利用したソフトウェア品質管理方法を提案した．次に，実際のプロジェクトに ESQUT を適用した結果，以下に示す効果を確認した．

1. 品質メトリクスの効果が明らかになった．各工程で，問題の早期発見，除去に活用できる．
2. 定期的な計測結果を定量化し，レビューを実施することによって，プログラム構造に良い影響を与えることがわかった．

3. テスト工程でも，設計，作成工程と同様に，定量化する．この情報と不具合情報を利用して，テストの状況を素早く，正しく把握できることがわかった．

ESQUT で，計測スケジュールと計測対象を設定すると，あとは自動的に，計測を行い，成果物の品質を定量化することができる．提案したモデルは，無理なく，簡単に，しかも精度よくリアルタイムに開発状況を把握し，改善活動に結びつけられることがわかった．

ESQUT の計測結果を最も効果的に使うための重要なポイントは，レビューと考えている．レビューをより効果的に実施するために，以下のような項目に関しても，検討する必要がある．

1. 計測結果のビジュアル化
   計測結果をより分かりやすく見せるために，グラフや集計などを利用したビジュアル化を推進する必要がある．リアルタイムに欲しい情報が取り出せる仕組みが必要である．
2. 使用者の視点
   開発者と管理者とでは，必要とする情報が違ってくる．各利用者の視点を考慮したデータのまとめ方を考える．
3. チェックの方法
   計測結果をより効果的に活用するために，チェックシート等を利用したレビュー方法を確立する．

付 録B　プログラム静的解析技術の効果的な活用方法

品質の高いソフトウェアを開発するための設計／コーディング技術，方法論，プロセスなど，今までに数多くの提案がなされている [77][21][79]．しかし，実際にはいまだ，品質の確保はテスト工程に依存しているのが現状である．テスト工程では，さまざまな観点からのテストやパスカバレッジの評価を行ない，テストの十分性を確認している．このような場合でも，完全に漏れのないテストを実施することは非常に困難である．

プログラムの静的解析では，問題が含まれている可能性のあるプログラム箇所，保守性や移植性で問題になりそうなプログラム箇所を指摘できる．コードレビューによる品質確認は効果的である．しかし，大規模なソフトウェアの場合，すべてのソースプログラムをレビューでカバーするには限界がある．ツールによる静的解析の有効性は，どんなに大規模なソフトウェアであっても，すべてのソースプログラムに対して機械的に解析を行なえることである．

ここでは，プログラム静的解析技術を利用するために必要な技術と，効果的な利用方法を示す．そして，適用事例とその評価について述べ，プログラム静的解析技術の有効性を明らかにする．また，静的解析技術を支援するツール，静的解析技術の限界についても考察する．

B.1　はじめに

従来から，プログラムを定量的に評価し，問題が入り込みやすいプログラムを早期に検出し，そのプログラムに対してコードレビューを行なうことによって，品質の作り込みを支援する活動は行なわれてきた [78]．

しかし，プログラムを定量的に評価した結果を開発者にフィードバックしても，実際にある程度出来上がってしまったソースプログラムを修正させる，という活動はかなり難しいのが現状である．これは，開発時間が確保できないなどの時間的な制約や，保守性や移植性などに対する重要性をなかなか理解してもらえないということが原因である．

また，コードレビューの有効性も多くの論文で述べられているが [77]，実際の開発現場で，定常的に実施されている部門は少ない．

プログラムの静的解析ツール（以後，静的解析ツール）は，通常コードレビューで指摘される問題（例えば，変数が初期化されていない，マジックナンバーが使われている，インデントがない等）の多くを指摘できる．

B.2 静的解析ツールの概要と問題点

静的解析ツールでは，表 B.1 に示すような種類の解析が行なえる．

表 B.1: 静的解析の種類と内容

No.	種類	内容
1	コード解析	潜在的なソフトウェア欠陥の検出
2	定量化	メトリクス計測
3	構造明確化	コーリング・ツリーの作成やグローバルデータの検出

ここでは，表 B.1 に示したコード解析について，概要と問題点について説明する．コード解析を行なう代表的なツールに lint やコンパイラの警告機能がある．図 B.1 に，lint での指摘内容の例を示す．

```
1:  void func ( )
2:  {
3:      short  s;
4:      long l = 100;
5:      s = l;
6:  }
=========
   assignment causes implicit narrowing
   conversion (5)
```

図 B.1: lint による指摘内容の例

図 B.1 で示したように，静的解析ツールでは，コンパイルはとおるが，動作上場合によってはエラーにつながるようなコードに対して警告を出す．静的解析ツールの機能のうち,「コード解析」機能に対する問題点を以下に示す．

1. 警告メッセージの数が多く，真に問題のあるコードを見逃してしまう．
2. 警告メッセージを抑制する場合には，ソースコードにコメントを埋め込んだり，ツール起動時のオプションを指定するなどの処理が必要であり，利用するために手間がかかる．
3. 警告メッセージの記述だけを見ても，内容がよく理解できない場合がある．

近年のコンパイラも，かなり高度な診断を行なえるようになってはいる．しかし，専用の静的解析ツールほどの診断を必ずしも行なうわけではないので，静的解析ツールの利用価値は十分に高いと考えられる．以後，静的解析のうち,「コード解析」に着目し，効果的な利用方法や適用結果について説明する．

B.3 静的解析技術の効果的利用方法

ソースコードの静的解析のために利用しているツールは，QA C/C++[1]というツールである．QA C/C++による静的解析結果の例を，図 B.2 に示す．

```
270:    if  ( *sts = NULL ) {
                ^

sample.c (270) ++WARNING++ : <=3= (3314) この制御式は
代入式 (assignment) です．—明示的に (explicit) 書いて，
ゼロに対するテストをするように．
 .... ( 中略 ) ....
280:
```

図 B.2: QAC/QAC++による警告メッセージ出力例

図 B.2 に示すとおり，QA C/C++では解析対象とするソースコードに警告メッセージを埋め込む形式で結果を出力する．開発担当者は，この解析結果から，修正すべきかどうかを検討し，修正が必要であればソースコードを修正する．そこで，図 B.2 のような解析結果とメトリクス計測結果を効果的に活用するために，以下のような活動を行なってきた．

1. 警告メッセージの重要度分類

   QA C/C++では，警告メッセージの数が約 500 種類ある．これらの警告メッセージの中には，コーディングスタイルやメトリクス値にもとづく警告も含まれる．ツールで用意されているすべての警告メッセージを対象にして解析を行なうと，ソースコードと警告メッセージの区別がつかなくなり，修正作業が困難になる場合が多い．このため，多くの解析事例から，チェックすべき重要度の高い 41 種類の警告メッセージ（明らかにエラーである，エラーの可能性がある，保守性／移植性に問題がある警告メッセージなど）を抽出した．抽出した警告メッセージの分類と代表的な例を，表 B.2 に示す．
2. 解析結果レポートの提供

   ツール解析の結果は，解析結果レポートとして参照しやすい形式で提供している．提供する情報は，対象としたソースコードから検出された”警告メッセージとその出現

[1]英国 PRL(Programming Research Ltd) 社で開発．日本では，(株) 東陽テクニカから販売されている．

表 B.2: チェックすべき重要度の高い警告メッセージの分類と代表例

No.	警告メッセージの分類	代表例
1	移植性（7 種類）	[1] 符号付きの値が暗黙のうちに符号無しに変換する [2] ポインタを整数に型変換（キャスト）する
2	コーディング 規約との不一致 （10 種類）	[3] switch 文に default がない [4] カッコ　なしで制御文を記述 [5] case 文が明示的な break 文で終わっていない
3	信頼性（18 種類）	[6] 制御式で代入式を使用（ if (*sts = NULL) …など） [7] 副作用の何もない文（代入文で '=' とすべきところを '==' としているなど） [8] 初期化されていない変数の使用 [9] 浮動小数点の式が等しいことを '==' や '!=' で比較する [10] マクロの定義（#define MAXIMUM 1024*16 等）
4	保守性（6 種類）	[11] 到達できない文 [12] 空の文 [13] C++ のキーワードを利用

数の一覧表"と"警告メッセージの出現箇所一覧表"である．これらの情報から，開発者は自分のソースコードにどのような問題が存在し，どの部分を修正すればよいか容易に把握できる．

3. 警告メッセージへの対応方法解説書作成

   静的解析ツールによって指摘されるソースコードの例と，その警告メッセージに対する対応方法を一覧表形式にまとめ，ドキュメントと Web 情報の両方で提供した．開発者は，検出された警告メッセージとそのメッセージへの対応方法を，この解説書や Web 情報から容易に確認できる．

4. ユーティリティツールの開発

   (2) と関連する内容であるが，解析結果レポートを作成するツールや，解析時に重要度のランクや特定の警告番号だけを指定して解析を行なうためのツールを提供し，開発部門で容易に静的解析ツールを利用できるようにした．また，メトリクス計測結果レポートを作成するツールも提供した．メトリクス計測結果は，重点的にレビューあるいはテストすべき対象の選定に活用する．

静的解析ツールを使用するための問題点を解決し，開発部門での静的解析ツールの定着化のためには，上述したような支援活動が重要である．

B.4　静的解析技術の適用事例

ここでは，約 60 件のプログラム静的解析結果に基づいて分析した結果を示す.

B.4.1　検出された警告数と検出密度

今までの品質解析サービスの結果から，サンプルとして 7 件のプログラムを対象に，検出された警告数と検出密度を評価した．評価の対象とした警告は，表 B.2 で示した重要度の高い 41 種類の警告である．評価結果を表 B.3 に示す.

表 B.3: 検出された警告数と検出密度

		プログラム							総検出数
		A	B	C	D	E	F	G	
解析時期		(デ)	(デ)	(結)	(総)	(総)	(総)	(総)	
検出数	移植性	47	35	471	300	454	931	589	2827
	コーディング規約との不一致	26	69	307	337	233	825	42	1839
	信頼性	37	64	386	203	304	419	139	1552
	保守性	3	5	295	373	17	635	9	1337
総検出数		113	173	1459	1213	1008	2810	779	7555
総コード行数		9K	40K	126K	36K	44K	87K	80K	
検出密度 (件/KLOC)		12.6	4.3	11.6	33.7	22.9	32.3	9.7	

注）解析時期　(デ):デバック、(結)結合テスト前、(総)総合テスト前

表 B.3 のプログラムは，それぞれ対象製品の違うものであり，関連はない．開発言語はすべて C である．表 B.3 の結果からも分かるとおり，警告メッセージの検出密度にはかなりバラツキがある．また，特定の警告メッセージが集中して検出される場合がほとんどであり，各警告メッセージが平均的に検出されることはない．総検出数がこれほど多くなった原因を以下に示す.

「移植性」では，以下の警告メッセージが全体（2827 件）の 88%であった.

- 符号付きの値が暗黙のうちに符号無しに変換されます.
- ポインタと整数型の間の型変換（キャスト）は処理系に依存します.

「Coding 規約との不一致」では，以下の警告メッセージが全体（1839 件）の 64%であった.

- else () を if ( ) 〜 else if ( ) 文に追加してください.
- 本文が 1 文しかなくても制御文には大カッコ　を付けてください.

「信頼性」では，以下の警告メッセージが全体（1552 件）の約 70%であった.

- 変数は，この時点では設定されていなかもしれません.

最後,「保守性」では，以下の警告メッセージが全体（1337 件）の約 84%であった.

- 空の文（”;”だけの文）があります.

これは，define 文の定義に，”;”が含まれていた（例：#define MAXDATA 7;）ことが大きな原因であった.

コンパイラのバージョンアップなどで，移植性に関連した警告箇所がエラーとなることもある．検出数が多くても正常に動作しているから対応しないというのではなく，集中して検出された特定の警告に対しても，変数の型宣言を厳密にしたり，define 文の定義を修正するなどして対応すべきである.

警告メッセージの検出数や種類は，開発する製品の特性ではなく，開発者の特性（言語の理解度や保守に対する意識の違い）に大きく依存していた．また，デバック，結合テストを終えた後でも，静的解析によって品質上問題のあるソースコードが検出されている．これは，レビューやテストによる品質評価だけでは，不十分であることを示している.

B.4.2　警告メッセージにもとづいたレビューの効果

表 B.3 に示した一つのプロジェクト（プログラム (C)）での，警告メッセージにもとづいたレビュー結果について説明する.

適用対象としたプログラムは，ハードウェアのデバイス制御を行なうプログラムである．開発担当者は 5 名であり，開発言語／規模は以下のとおりである.

- 開発言語：C
- ファイル数：67
- モジュール数：1511
- コード行数：126KLOC(LOC:Lines of code)

静的解析は，約 2 週間の間隔をあけて 2 回実施した．静的解析を適用した工程は，各開発担当者がデバック作業を完了した段階と結合テスト中の 2 回である.

(1) 指摘件数と修正件数

デバッグ作業完了後に実施した静的解析結果の例を，図 B.3 に示す．この表は，B.3 節 (2) で説明したツールを利用して出力したものである．表 B.3 では，検出された総警告メッセージ数は 1459 件であったが，これだけの件数を確認するための工数を確保できなかったため，集中的に検出された特定の警告メッセージを除き，さらに重要度の高い警告メッセージだけに絞り込んだ．その結果検出された総警告メッセージ数は 250 件であり，警告メッセージの検出密度は，1.98 件/KLOC であった．

[QAC analysis report (Directory)]
ディレクトリ名 : /home/ssel/Project/WB
編集日 : 97.10.13(Mon)

警告番号	出現頻度	警告メッセージ
434	2	宣言されていない識別子に対して'extern int TEXT;' 宣言があるものとしました。
3611	5	char'型から、または'char'型への暗黙の変換の結果は、その'char'が符号付きか否かに依存します。異なる'char'型を式の中に混ぜないようにすることがより安全です。
3614	7	マクロTEXTが再定義されています。
		:
3334	3	この'TEXT'の宣言は、もっと広域(global)な宣言を隠しています。これは非常に混乱する方法です。また、このやり方は原因を追跡するのが非常に難しいバグを引き起こしかねません。
3314	6	この制御式は代入文です。ー明示的になるように記述しなおして、ゼロかどうかのテストをするようにして下さい。
3328	2	符号無しの値は決して負の値になりません。ーこのテストは正しく評価されません。
3343	1	比較の被演算数の片側で論理否定が行われています。
		:

図 B.3: 検出された警告メッセージと出現頻度一覧

開発担当者が静的解析結果を参照しながらコードレビューを行った．コードレビューの結果，修正対応された件数を表 B.4 に示す．

コードレビューの結果，修正対応された件数は 88 件であり，検出された警告メッセージに対する修正率は約 35%であった．また，担当者それぞれがレビューに要した時間は 1

表 B.4: 修正対応された件数

No.	警告メッセージの種類	検出数	対応数
1	移植性	36	9
2	コーディング規約との不一致	97	42
3	信頼性	40	10
4	保守性	77	27
	合計	250	88

〜2 時間であり，かなりの短時間に効率的なレビューが行なえたといえる．

修正対応された警告メッセージの中には，明らかにエラーであるコードも含まれていた．代表的なエラーの例を以下に示す．

1. 比較の演算数の片側で論理否定が行なわれていた

```
修正前：if ( !XXX == YYY ) { ---
修正後：if ( XXX != YYY) { ...
```

2. 条件ループ文の記述ミス

```
修正前：for ( i < 0 ; i < REC_MAX ; ++i ) { ...
修正後：for ( i = 0 ; i < REC_MAX ; ++i ) { ...
```

3. 何の副作用もない文が含まれている（動作上エラーにはならない）

```
修正前：100:  xxx->yyy = zzz;
        101:            + ( ZZZ
        102:                << XXX);
修正後：100:  xxx->yyy = zzz;
        101:          /* +( ZZZ
        102:              << XXX) */
```

(2) 警告メッセージの時系列的な傾向

デバッグ終了時点での 1 回目の静的解析のあと，約 2 週間後に静的解析を再度実施した．ソースコードの増加の傾向を表 B.5 に示す．

2 回目の静的解析結果では，表 B.5 に示すとおり，ソースコードの行数は増えていたが，1 回目で対応した警告メッセージは検出されなかった．これは，開発者がレビュー作業をとおして，プログラミングスタイルの定石や C 言語の特性を理解したことによると考えられる．

表 B.5: ソースコードの増加傾向

No.	項目	1 回目	2 回目	増加分
1	ファイル数	67	69	2
2	モジュール数	1511	1536	25
3	コード行数	126K	128K	2438 行

B.5 プログラム静的解析の効果分析

ここでは，B.4 節の結果にもとづいて，以下の 4 つの観点から分析・考察する．

- 定量的分析
- 定性的分析
- 静的解析を支援するツールの能力
- 静的解析技術の限界

B.5.1 定量的分析

ここでは，工数の観点から定量的な分析を行なう．

一般に，コーディング工程で検出されたエラーを修正する作業工数を 1 とすると，テスト工程，運用工程に持ち越したときの相対的な工数は，表 B.6 のような値になるといわれている [79]．

表 B.6: エラー修正に要するコスト

工程	設計〜コーディング	テスト	運用
1 件当たりのエラー修正コスト	1	1.5 〜 8	5 〜 60

今回の適用事例では，250 件の指摘に対して，88 件の修正が行なわれている．この修正はデバッグ作業（コーディング工程の作業とみなす）で行なわれている．88 件の修正内容をチェックした結果，直接エラーに関わるものは約 30% （26 件）と見積ることができた．デバッグ作業で検出できた 26 件のエラーを，テスト工程で検出／修正するためには，最大 8 倍の工数がかかることになる．しかも，それらのすべてがテスト工程で検出／修正されるとは限らない．もし，運用段階までエラーを持ち越した場合には，そのエラーの修正工数は，さらに増加することが容易に理解できる．

次に，コードレビューを実施する場合と比較して検討する．

コードレビューの重要性は広く認識されており，古くよりさまざまなソフトウェアレビューの方法論が提案されている [77][83]．しかし，少人数による開発や厳しい納期の制約のもとで十分に効果的なレビューを実施することは難しい．

論文 [77] によると，140KLOC のシステムに対して，18 回のレビュー（インスペクション）を実施し，11.5 件/KLOC の指摘ができたという報告がある．このレビューは，ソースコード全体の 29%に相当するものであった．指摘できた内容は保守性やコーディングスタイルに関する項目，異常ケースの考慮抜けなどである．

今回の適用事例では，最重要警告メッセージだけを対象としていたため，保守性に含まれる指摘（コーディングスタイルに関する指摘，マジックナンバー／データの埋め込み，長すぎる関数／複雑過ぎるロジックなどに関する指摘）の多くは，静的解析結果には含まれていない．このような指摘も含めて静的解析を実施すると，論文 [77] の指摘率に近い値になる．すなわち，人間が指摘できるレビュー内容のかなりの部分を，静的解析ではカバーできると考えられる．もちろん，人間ができる処理ロジックの論理的ミスの指摘はできないが，静的解析ではすべてのソースコードを対象に機械的に解析が可能というメリットがある．この静的解析技術とレビュー技術を組み合わせて活用することは，コーディング工程で効率よく，しかも漏れのない検証を行なうための有効な手段である．

B.5.2 定性的分析

プログラム静的解析ツールを利用することの定性的な効果についてまとめる．

1. 保守性／移植性の向上

   今回の適用事例では，保守性／移植性に関する指摘のうち，以下の項目については修正が行なわれている．このように，静的解析を行なうことにより，保守性／移植性の向上にも有効である．

   - 関数の戻り値を受け取っていない関数呼出し部分の修正
   - 定義されたが利用されていない変数の削除
   - 使用されていないラベルの削除
   - 関数プロトタイプを使用するように修正
   - C++の予約語を変数として使っていたものを修正

2. 教育効果

   今回の適用事例では，1 回目の静的解析から 2 回目の静的解析までに，新しく混入された警告メッセージはなかった．これは，開発者がプログラミングスタイルの定石や C 言語の特性を理解したことによると考えられる．

   近年，少人数による開発や厳しい納期の制約などで熟練者が経験の浅いソフト技術者のソースコードをレビューする機会が減少傾向にある．その結果，プログラミング技術が伝承されず，開発者の能力に大きな差が生じることになる．静的解析技術

を利用して，プログラミングの定石や言語の特性を理解させることは，開発者の能力を引き上げることにもつながる．

3. 全ソースコードの検証

人間によるコードレビューの多くの場合と異なり，静的解析は，全ソースコードを機械的に検証できる．例えば，ツールを利用すると容易に検出できるが，レビューでは検出できない場合の事例を紹介する．我々は，社内の「ソフトウェア品質保証教育」の中で，図 B.4 に示したソースコードを提示してコードレビューを行なわせている．

```
                :
switch ( type ) {
    case 3:
        type_tb [0] = 3;
        from_tb [0] = 1;
        break;
            :
    defalut:
        irnt = 13;
        break;
}
                :
```

図 B.4: 教育用のソースコード

明らかにこれは，”default（デフォルト）”のスペルミスであり，”default” とすべきところが”defalut”となっている．人間がチェックすると，思い込みや漏れなどがあるため，これを指摘できない場合がある．また，図 B.4 の場合では，テスト工程でデフォルト処理を通るデータが生成され，テストされるという可能性が 100%あるわけではない．すなわち，エラーが検出されないままリリースされることもありうる．大規模なソースコードを対象とするときには，人間の能力でコードレビューできる範囲／内容，および漏れのないテストケースの生成には限界がある．静的解析で全ソースコードに対する安全性を検証することは，品質を保証するための最低限のマナーである．

B.5.3 静的解析を支援するツールの能力

静的解析を支援するツールには，以下の 3 つのツールがあると考えられる．

1. コンパイラの警告機能
2. lint

3. 静的解析に特化した市販ツール

ここでは，品質解析サービス活動の中でしばしば見つかった，直接エラーにつながる危険なソースコードに対する上記ツールの検出能力を調査した．使用したツールとオプションを表 B.7 に示す.

表 B.7: 使用したツールとオプション

No.	分類	ツール名	オプション
1	コンパイラ	gcc-2.7.2.3	-Wall
2	lint	lint SunOS4.1 版	-a -b -c -h
3	専用ツール	QAC	

次に，静的解析技術を使って検出したい危険なソースコードのパターンを 13 個抽出した．これは，表 B.3 の代表的な例の番号に対応する．これに，評価の順番が未定義の場合のパターンを 1 個追加した.

1. 符号付きの値が暗黙のうちに符号なしに変換する
2. ポインタを整数に型変換（キャスト）する
3. switch 文の default 抜け/default 記述ミス
4. カッコ　なしで制御文を記述

```
if ( x == 1)  return(value);
 :
```

5. case 文の break; 抜け
6. 制御式で代入式を使用

```
if ( *sts = NULL ) {
  :
```

7. 副作用の何もない文

```
i == 3;
 :
```

8. 初期化されていない変数の使用
9. 浮動少数点の式が等しいことを '==' や '!=' で比較する
10. マクロの定義エラー

```
#define  MAXDATA  1024*16
#define  MAXDATA  100;
```

11. 到達しない文

```
return (value);
if (a == b) {
 :
```

12. 空の文
13. C++のキーワードを利用
14. 評価の順番が未定義の場合

```
i = 0;
func (i , ++i );
 :
```

これら 14 パターンのサンプルプログラムを作成し，表 B.7 で示したツールを使用して，静的解析を行った．その結果を，表 B.8 に示す．○印は，警告されたことを示している．

表 B.8: 各静的解析ツールでの結果

分類	パターン	gcc	lint	QAC
移植性	[1]			○
	[2]		○	○
コーディング規約との不一致	[3]			○
	[4]			○
	[5]			○
信頼性	[6]	○		○
	[7]	○	○	○
	[8]		○	○
	[9]			○
	[10]			○
保守性	[11]		○	○
	[12]			○
	[13]			○
コーディング規約との不一致	[14]		○	○

表 B.8 の結果から，静的解析の能力は専用ツールが一番高いことが分かる．特に，プログラミングのスタイルに関係する部分，移植性に関係する部分の解析能力が高い．

各開発部門では，コーディングガイドなどを用意している．そのガイドを陳腐化させないために，専用の静的解析ツールを利用し，ガイドにしたがったプログラミングが行なわれているかどうかチェックすることは有効である．また，おこしやすい警告エラーをフィードバックしてコーディングガイドに付加することは，技術の横展開と向上につながる．

B.5.4 静的解析技術の限界

動的解析ツールは，検出や原因究明が難しいメモリアクセスに関するエラーを自動的に検出する．動的メモリツールでエラーを検出できる簡単なサンプルプログラムを，図 B.5 に示す．図中の (A)，(B) の部分が，動的メモリツールで検出できるエラーである．

```
        #include <stdio.h>
        #include <string.h>
        #include <malloc.h>

        main()
        {
            char *message1 = "Hello World.\n";
            char *message2 = "Hello World Again.\n";
            char *s = NULL;
            char buf[128];

            printf("%s\n",message1);

            s = malloc(strlen(message1) * sizeof(char));
            if (s == NULL) {
                printf("malloc() failed\n");
                exit(
            }

A --->  printf("%s\n",s);     /* Read un-initialized memory*/
B --->  strcpy(s,message1);   /* Write beyond memory bound */

            memcpy(buf,s,(strlen(s)+1)*sizeof(s));
                                  /* Read beyond memory bound */
            printf("%s\n",buf);
                      :
```

図 B.5: メモリアクセスエラーを含むサンプルプログラム

このサンプルプログラムに対して，表 B.7 で示した各ツールを使用して静的解析を実施した．その結果，静的解析ツールでは，図 B.5 で示されているメモリアクセスに関するエラーは検出できなかった．

動的解析ツールと静的解析ツールの役割が違うので，この結果は当然かもしれない．しかし，図 B.5 のようなサンプルプログラムの場合には，静的解析レベルでもメモリアクセスに関するエラーの検出はある程度できると考えられる．例えば，図 B.5 の (A) の部分は，

malloc() で領域が確保されたあと，初期化されないまま読み出そうとしている．このような場合には，領域が確保された後，代入されたかどうかをチェックすることによって，静的解析でも未初期化メモリからの読み込みを検出できる．また，図 B.5 の (B) の部分は，メモリの割り当て範囲を越える書き込みである．この場合も，変数のサイズがチェック（特定）できるので，静的解析で検出できると思われる．

このように，現状の静的解析技術では，動的なメモリアクセスに関する解析能力が弱い．動的な解析と同等のレベルまでは当然できないであろうが，静的解析技術の利用によって，ある特定のパターンのときは，メモリアクセスに関するエラーを検出できると考えられる．

静的解析技術に関する適用結果と考察から，以下に示す手順で品質保証することを提案する．

- コーディング工程では，静的解析技術とレビュー技術を組み合わせて活用し，全ソースコードに対する検証を行なう．
- テスト工程では，さまざまな観点から漏れのないテストケースを作成する．そのテストの実施時に，動的解析技術を利用し，メモリアクセスに関するエラーがないことを検証する．

B.6 まとめ

「プログラム静的解析技術の効果的な活用方法」について述べた．従来，プログラム静的解析技術が重要であることは認識されていると思われるものの，実際の開発部門内で効果的に利用されている場面は少なかった．

ソフトウェアの開発は，開発場所の分散化が進み，規模もますます増大する傾向にある．このような背景を考えると，ソフトウェアの品質を効率的に保証するためには，積極的に機械化を推進する必要があると考えている．幸いなことに，品質保証のためのツールは，かなり質の高いものが揃ってきている．ただし，それを使いこなすまでには，それなりの時間とノウハウが必要となる．第三者的な組織で有効なツールや手法を選定し，ノウハウを蓄積することは，開発部門の機械化を支援するための有効な手段である．

謝辞

本論文をまとめるにあたり，終始あたたかいご指導を賜りました電気通信大学　大学院情報システム学研究科の大須賀昭彦教授に厚く感謝の意を捧げます．また親切なご教示を賜りました電気通信大学 大学院 情報システム学研究科の田中健次教授に心から感謝申し上げます．本研究を進める上で多くのご指導とご鞭撻を賜りました電気通信大学 大学院情報システム学研究科の田原康之准教授，中川博之助教に感謝の意を表します．

審査を快く引き受けてくださいました電気通信大学 大学院 情報システム学研究科の渡辺俊典教授，鈴木和幸教授に感謝申し上げます．先生方には，論文のまとめ方や技術の評価方法などに関して多大なご指導をいただきました．

本研究は株式会社東芝において，多くの方々のご指導とご協力を得て行ったものに基づいています．東芝グループ内でソフトウェアプロセス改善活動を一緒に推進してきた多くの方々に感謝致します．また，日々，活発な議論をさせていただき，また，さまざまなご支援とご配慮を頂きました株式会社東芝 ソフトウェア技術センター プロセス・品質技術開発担当の先輩，同僚，後輩の方々に心から御礼申し上げます．

最後に，日々の研究活動を心身両面にわたって支えてくれた家族に心から感謝します．

参考文献

[1] W. W. Gibbs: Software's Chronic Crisis, Scientific American, September, pp.86-95, 1994.

[2] 高橋 秀俊: ソフトウェア危機 （The Software Crisis） , 情報処理学会会誌 Vol.10, No.6,pp.373-374, 1969.

[3] 井上克郎, 松本健一, 飯田元. :ソフトウェアプロセス, 共立出版, pp.1-5, pp.3, pp.27, pp.51, pp.199,2000.

[4] Software Hell, Business Week, http://www.businessweek.com/1999/99_49/b3658015.htm, Dec.6,1999

[5] W. Royce: Managing the Development of Large Software Systems, Proc. 9th International Conference on Software Engineering, CA, pp.328-338, April 1987.

[6] A. Fuggetta, A. Wolf (Ed): Software Process, International Conference on Software Engineering, Proceedings of the Conference on The Future of Software Engineering, pp.25-34, 2000.

[7] L. Osterweil: Software Process Are Software Too, Proc. 9th International Conference on Software Engineering, Monterey CA, pp.2-12, April 1987.

[8] 田中康 : プロセス改善のための成果物観点によるソフトウェアプロセスモデル化手法, NAIST-IS-DT0261016(博士論文), 2005.

[9] 坂本啓司 : 実プロジェクトデータの分析に基づくソフトウェアプロセス改善と品質向上の研究, NAIST-IS-DT9861009(博士論文), 2000.

[10] 飯塚悦功 : プロセス改善の本質, 上海軟件創新フォーラム プロセス改善大会, 2010.

[11] 艸薙匠他 : 組織の開発プロセスの（再）設計による改善活動, 第 15 回ソフトウェア生産における品質管理シンポジウム, 1995.

[12] 松原友夫・乗松聡: 日本における CMM 導入を考える, 共立出版 bit, Vol.30, No3, pp 2-8, 1998.3.

[13] 情報処理学会会誌「情報処理」: ソフトウェア管理技術の最新動向を探る, Vol.44 No.4, 2003.4.

[14] CMM/CMMI Maturity Profile,http://www.sei.cmu.edu/sema/profile.html

[15] 小川俶子：特別企画 ソフトウェアプロセスアセスメントの現状「プロセス改善活動の中で感じたことと問題提起」, 標準化と品質管理, Vol.63 2010 No.2, 2010.

[16] 金原洋子： SPI 全社展開について ～前回の失敗事例を教訓とした実践事例報告～, SPI Japan 2009, http://www.jaspic.org/event/2009/SPIJapan/session1A/1A2.pdf, 2009.

[17] Fukuyama, S., Miyamura, S., Takagi, H. and Tanaka, R. : A Software Process Improvement Support System: SPIS, ICISE Trans, Information and Systems, Special Issue on Knowledge-Based Software Engineering, Vol. 4, No. 10, pp. 747–756 (2000).

[18] Robert McFeeley, IDEAL : A User's Guide for Software Process Improvement, Handbook CMU/SEI-96-HB-001, (1996).

[19] 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, pp. 529–541 (2001).

[20] Gargi Keeni, The Evolution of Quality Processes at Tata Consultancy Services, IEEE Software, vol. 17, no. 4, pp. 79–88 (2000).

[21] Hideto Ogasawara, Atsushi Yamada, Michiko Kojo: Experiences of Software Quality Management Using Metrics through the Life-Cycle, 18th ICSE, pp. 179–188 (1996).

[22] Hideto Ogasawara, Minoru Aizawa, Atsushi Yamada: Experiences with Program Static Analysis, Fifth International Software Metrics Symposium (1998).

[23] Takeshi Tanaka, Minoru Aizawa, Hideto Ogasawara, Atsushi Yamada: Software Quality Analysis & Measurement Service Activity in the Company, 20th ICSE, pp. 426–429 (1998).

[24] Hideto Ogasawara, Takumi Kusanagi, Mikako Arami: Process Improvement Activities by High Quality Software Creation Support Virtual Center, 2WCSQ(The Second World Congress for Software Quality) (2000).

[25] 2009 年版 組込みソフトウェア産業実態調査報告書 〜経営者及び事業責任者向け調査〜, 経済産業省 商務情報政策局 情報処理振興課, 2009.7(平成 21 年 7 月).

[26] 2010 年版 組込みソフトウェア産業実態調査報告書 〜事業責任者向け調査〜, 経済産業省 商務情報政策局 情報処理振興課, 2010.6(平成 22 年 6 月).

[27] Baetjer,Jr.,H., : Software as Capital, IEEE Computer Society Press, 1998, pp.85.

[28] IEEE Std 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology, IEEE Computer Society, 1990.

[29] ロジャー S. プレスマン著, 西 康晴／榊原 彰／内藤裕史 監訳, 古沢聡子／正木めぐみ／関口 梢 翻訳 : 実践ソフトウェアエンジニアリング 〜ソフトウェアプロフェッショナルのための基本知識〜, 日科技連出版社, 2005.

[30] Paulk, M., et al., : Capability Maturity Model for Software, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1993.

[31] W.S.Humphrey, Managing the Software Process, Addison-Wesley, 1989.

[32] 飯塚悦功編 : ソフトウェアの品質保証, 日本規格協会, 1992.

[33] SPA(ISO/IEC TR 15504-1,2,3,4,5,6,7,8,9), 1998.

[34] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber : Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, (1993).

[35] Mark C. Paulk, Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis, and Marilyn W. Bush : Key Practices of the Capability Maturity Model, Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-25, DTIC Number ADA263432, (1993).

[36] Mary Beth Chrissis, Mike Konrad and Sandy Shrum : CMMI : Guidelines for Process Integration and Product Improvement, Addison-Wesley (2003).

[37] CMU/SEI-2006-TR-004, Performance Result of CMMI-based Process Improvement, CMU/SEI, 2006.

[38] James Herbsleb, et.al., Software Quality and the Capability Maturity Model, CAMC Vol1.40, No.6, 1997.

[39] CMMI for SCAMPI Class A Appraisal Results - 2011 Mid-Year Update, Software Engineering Institute / Carnegie Mellon University,Pittsburgh, PA 15213, September 2011.

[40] 板橋吉徳，中山貴史，浅井理恵子，吉村宏之：全社ソフトウェアプロセス改善の取り組み，Panasonic Technical Journal Vol.56 No.3 Oct.2010.

[41] 大場充, 堀田勝美, 松瀬健司： ソフトウェアプロセス改善と組織学習 ～CMM を毒にするか？ 薬にするか？～, ソフト・リサーチ・センター, 2003.

[42] 「CMM 偏重は見直す」経済産業省が政府調達の方針を転換，NIKKEI COMPUTER 2001.8.13.

[43] 乗松聡： 問題解決重視とモデル重視 ～組織に合ったプロセス改善モードを探す～, SEPG Japan 2004.

[44] 小笠原秀人, 藤巻昇, 艸薙匠. 田原康之, 大須賀昭彦: 大規模組織におけるソフトウェアプロセス改善活動の適用評価 ～10 年間の実践に基づく考察, 情報処理学会論文誌, Vol.51, No.9, pp.1805～1815, Sep. 2010.

[45] Hideto Ogasawara, Atsushi Yamada, Takumi Kusanagi, Mikako Arami: How to Effectively Promote the Software Process Improvement Activities in a Large-Scale Organization, ECSQ (2002).

[46] Hideto Ogasawara , Noboru Fujimaki, Shoichi Kojima: The Evolution of Software Process Improvement Activities in a Large-Scale Organization, CONQUEST (2004).

[47] Hideto Ogasawara, Takumi Kusanagi, Noboru Fujimaki: Practical Approach to Evolve SPI Activities in a Large-Scale Organization, 3WCSQ (2005).

[48] Hironobu Aoki,Tetsuro Moriya,Hideto Ogasawara: Workshop Driven Software Process Improvement, 4WCSQ (2008).

[49] ITA スキルマネジメント WG : IT ベンダによる ETSS 適用事例報告 ～ETSS を活用したスキルマネジメント～, SEC journal Vol.5 No.1 Mar.2009, (2009).

[50] Minoru Aizawa, Yasutaka Shirai, Masahiko Ohki, Noriyuki Nagai, Reika Satoh, Akihiro Sugiyama: An Approach to Improvement of Software Project Management by Utilizing Data from SCM System, 3WCSQ (2005).

[51] Ogasawara, H., Ishikawa, T. and Moriya, T. : Practical approach to development of SPI activities in a large organization : Toshiba's SPI history since 2000, ICSE, pp. 595–599 (2006).

[52] 小森真紀, 藤巻昇:ソフトウェア品質保証活動を推進できる人材の育成 〜SQAG 教育コースの開発と実践〜, ソフトウェアプロセス改善シンポジウム (SPI Japan), 2009.

[53] 小笠原秀人, 青木裕伸, 鷲見毅:ワークショップをベースにしたプロセス改善活動 〜問題解決を出発点とするために〜, ソフトウェアプロセス改善カンファレンス (SPI Japan), 2006.

[54] 丸屋宏二, 艸薙匠:改善文化を創る SPI 〜改善意識中心アプローチ〜, ソフトウェアプロセス改善カンファレンス (SPI Japan), 2007.

[55] Kenneth M. Dymond: A Guide to the CMM -Understanding the Capability Maturity Model for Software-, Process Transition International, Inc., 1998.

[56] Sugahara, K., Ogasawara, H., Aoyama, T. and Higashi, T. : Status of SPI Activities in Japanese Software - A view from JASPIC -, Software Process Workshop at Beijing (2005)

[57] Takuro Iida, Hideto Ogasawara, Takumi Kusanagi: Improve to Meet Your Organizational Needs using CMMI Continuous Representation - Good-bye, Level Hunting! -, 4WCSQ (2008).

[58] Tetsuro Moriya, Atsushi Yamada, Takashi Ishikawa: Enterprise Level Deployment of Quantitatively Project Management, SEPG 2009 North America (2009).

[59] 小笠原秀人，小島昌一 : 定着を重視したプロセス改善活動，情報処理，Vol. 44, No. 4, pp. 10–16 (2003).

[60] 艸薙匠, 猪野仁, 石川隆: ソフトウェアプロセス改善活動, 東芝レビュー Vol.61 No.1, 20006.

[61] 小笠原秀人，原田明宏，杉山昭洋，吉崎浩二 : STMpj によるソフトウェア品質改善の推進 (3):静的解析で得られたデータの分析結果，情報処理学会第 58 回全国大会 (1999).

[62] 藤巻昇, 丸屋宏二, 小笠原秀人 : 不具合管理ツールによる不具合管理プロセス改善活動, 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, 2001.

[63] 竹内 朝一: 統計的手法による不具合予測モデル構築と活用 ー定量的プロジェクト管理の定着とさらなる発展を目指してー, JASPIC 夏のソフトウェアプロセス改善セミナー, 2010.

[64] 守屋哲朗, 山田淳, 石川 隆: 東芝グループにおける定量的プロジェクト管理の展開, 東芝レビュー Vol.64 No.10, 2010.

[65] M.Hirayama, et.al., Practice of quality modeling and measurement on software lifecycle, Proceedings of the 12th ICSE, pp98-107,March 1990.

[66] A.Yamada, et.al, Quantitative analysis method of software design characteristics for quality improvement, Proceeding of 2nd EOQC, May 1990.

[67] T.J.McCabe, A Complexity Measure, IEEE Trans. on Software Eng., Vol.SE-2, No.4, 1976.

[68] M.H.Halstead, Elements of Software Science, Elsevier NorthHolland Pub., N.Y., 1977.

[69] Richard Bache, et.al, Software Metrics for Product Assessment, McGraw-Hill BOOK COMPANY, London, 1994.

[70] V.R.Basili and R.W.Selby, Comparing the Effectiveness of Software Testing Strategies, IEEE Trans. Software Eng., Vol. SE-13, No.12, pp.1278-1296, 1987.

[71] V.R.Basili and R.W.Reiter,Jr, Evaluation Automatable Measures of Software Development, IEEE, Tutorial on Models and Metrics for Software Management and Engineering", pp280-289, 1980.

[72] V.R.Basili,R.W.Selby and D.H.Hutcheus, Experimentation in Software Engineering, IEEE Trans., Software Eng., Vol.SE-12, No.7, pp733-743, 1986.

[73] C.Jones, Programming Productivity, McGraw-Hill, NewYork, 1986.

[74] George Stark and Robert C.Durst, Using Metrics in Management Decision Making, IEEE COMPUTER, pp.42-48, September, 1994.

[75] Rajiv D. Banker, Robert J. Kauffman, Charles Wright, and Dani Zweig, Automating Output Size and Reuse Metrics in a Repository-Based Computer-Aided Software Engineering (CASE) Environment, IEEE Trans. Software Eng., Vol. 20, No.3, pp.169-187, 1994.

[76] 会沢他：プログラム解析・計測ツールを利用したソフトウェア品質向上支援活動, 日科技連, 第 17 回ソフトウェア生産における品質管理論文集, 1997.

[77] 猪野他：効率的レビュープロセス設計方法について, 情処学ソフトウェア工学研報, 112-3, 1996.

[78] 小笠原秀人：メトリクスを利用したソフトウェア品質管理の提案と実践，情処ワークショップ論文集 Vol.95, No.1, pp.27-34, 1995.

[79] 森口繁一監修：ソフトウェア品質管理ガイドブック，日本規格協会，1990.

[80] Programming Research Limited ： QA C 3.2 User Guide & Reference Manual, 1995.

[81] LesHatton： Safer C, McGRAW-HILL, 1995.

[82] Thomas Plum： C Programming Guidelines, PLUMHALL, 1989.

[83] Robert Grandy and Tom Van Slack： Key Lessons in Achieving Widespread Inspection Use, IEEE Software, Volume 11, Number 4, 1994.

[84] スティーブ・サミット著, 北野欽一訳：C プログラミング FAQ, トッパン, 1997.

[85] 古賀国秀, 山元和子: ソースコード静的解析技術, 東芝レビュー Vol.64 No.4, 2009.

[86] 久連石圭, 河村透, 小笠原秀人, 佐々木愛美: テスト管理システムを用いたテスト状況の可視化方法の提案, 情報処理学会第 73 回全国大会, 2011.

[87] 小笠原秀人, 河村透, 久連石圭, 佐々木愛美: 導入容易なテスト管理システムの開発とその適用効果, 第 31 回ソフトウェア・シンポジウム (SS201), 2011.

研究業績

学術誌論文

1. 小笠原秀人, 藤巻昇, 艸薙匠, 田原康之, 大須賀昭彦: 大規模組織におけるソフトウェアプロセス改善活動の適用評価—10 年間の実践に基づく考察, 情報処理学会論文誌, Vol.51, No.9, pp.1805-1815, Sep. 2010.

国際会議・ワークショップ論文

1. Hideto Ogasawara, Atsushi Yamada, Michiko Kojo: Experiences of Software Quality Management Using Metrics through the Life-Cycle, ICSE 1996(pp.179-188), 1996.

2. Hideto Ogasawara, Minoru Aizawa, Atsushi Yamada: Experiences with Program Static Analysis, Fifth International Software Metrics Symposium, 1998.

3. Takeshi Tanaka, Minoru Aizawa, Hideto Ogasawara, Atsushi Yamada: Software Quality Analysis Company, ICSE 1998(pp.426-429), 1998.

4. Hideto Ogasawara, Takumi Kusanagi, Mikako Arami: Process Improvement Activities by High Quality Software Creation Support Virtual Center, 2WCSQ(The Second World Congress for Software Quality), 2000.

5. Hideto Ogasawara, Atsushi Yamada, Takumi Kusanagi, Mikako Arami: How to Effectively Promote the Software Process Improvement Activities in a Large-Scale Organization, ECSQ 2002(pp.124-134), 2002.

6. Toyohiro Kanayama, Hideto Ogasawara, Hiroshi Kimijima: Quality Control Techniques for Constructing Attractive Corporate Websites: Usability in Relation to the Popularity Ranking of Websites, ECSQ 2002(pp.53-62), 2002.

7. Hideto Ogasawara , Noboru Fujimaki, Shoichi Kojima: The Evolution of Software Process Improvement Activities in a Large-Scale Organization, CONQUEST 2004, 2004.

8. Kouichi Sugahara, Hideto Ogasawara, Teruyuki Aoyama, Tetsuya Higashi: Status of SPI Activities in Japanese Software - A View from JASPIC, ISPW 2005(pp.498-506), 2005.

9. Hideto Ogasawara, Takumi Kusanagi, Noboru : Practical Approach to Evolve SPI Activities in a Large-Scale Organization, 3WCSQ (The Third World Congress for Software Quality), 2005.

10. Hideto Ogasawara, Takashi Ishikawa, Tetsuro Moriya: Practical approach to development of SPI activities in a large organization: Toshiba's SPI history since 2000, ICSE 2006(pp.595-599), 2006.

11. Takuro Iida, Hideto Ogasawara, Takumi Kusanagi: Improve to Meet Your Organizational Needs using CMMI Continuous Representation - Good-bye, Level Hunting! -, 4WCSQ (The Fourth World Congress for Software Quality) , 2008.

12. Hironobu Aoki,Tetsuro Moriya,Hideto Ogasawara: Workshop Driven Software Process Improvement, 4WCSQ (The Fourth World Congress for Software Quality) , 2008.

国内研究会論文

1. 小笠原秀人, 平山雅之, 宗近修久, 岡安二郎, 深谷哲司, 三原幸博: ソフトウェア品質保証システム ESQUT ～仕様書段階における評価技術～, 情報処理学会第 42 回 (平成 3 年前期) 全国大会, 1991.

2. 小笠原秀人, 平山雅之, 岡安二郎, 深谷哲司: 自然言語で記述された仕様書の品質評価技術, 信学技報 Vol.91 No.35, 1991.

3. 小笠原秀人, 三原幸博, 平山雅之, 岡安二郎, 深谷哲司: ソフトウェア仕様書品質評価支援システム ESQUT-SPC, 第 11 回ソフトウェア生産における品質管理シンポジウム,1991.

4. 小笠原秀人, 平山雅之: ソフトウェア品質評価支援ツール ESQUT の適用と評価, 情報処理学会研究報告, 情処研報 Vol.92, No.79,1992.

5. 小笠原秀人, 山田淳, 古城路子: メトリクスを利用したソフトウェア品質管理の提案と実践, サマーワークショップ・イン・立山, 情処ワークショップ論文集 Vol.95, No.1,1995.

6. 会沢実, 小笠原秀人, 山田淳: プログラム静的解析ツールを利用したソフウェア品質向上支援活動, 第 16 回ソフトウェア生産における品質管シンポジウム,1997.

7. 小笠原秀人, 会沢実, 山田淳: プログラム静的解析の適用事例と効果分析, ソフトウェア・シンポジウム 1998,1998.（最優秀論文賞受賞）

8. 小笠原秀人, 原田明宏, 杉山昭洋, 吉崎浩二: STMpj によるソフトウェア品質改善の推進 (3):静的解析で得られたデータの分析結果, 情報処理学会第 58 回全国大会,1999.

9. 杉山昭洋, 原田明宏, 中村泰博, 小宮康治, 小笠原秀人, 吉崎浩二: 光学測定機開発におけるソフトウェア品質改善フレームの構築と実施例, 光計測シンポジウム,1999.

10. 青山幹雄, 小笠原秀人, 金谷延幸, 中所武司, 佃軍治, 中谷多哉子, 深澤良彰, 中島震: 次世代ソフトウェアアーキテクチャへの展望, 第 2 回戦略的ソフトウェア研究開発シンポジウム,2000.

11. 艸薙匠, 荒見美香子, 小笠原秀人: CMM 活動: アンケート方式によるソフトウェアプロセス診断改善活動, ソフトウェア・シンポジウム 2000, 2000.

12. 小笠原秀人: ソフトウェア開発プロセスの評価と改善, ウィンターワークショップ in 伊豆 (情報処理学会),2001.

13. 藤巻昇, 丸屋宏二, 小笠原秀人: 不具合管理ツールによる不具合管理プロセス改善活動, 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, 2001.

14. 小笠原秀人, 山田淳, 艸薙匠, 荒見美香子: 階層的な SEPG の構築による SPI 活動の実践事例, ソフトウェア・シンポジウム (SS2002), 2002.

15. 田村朱麗, 新井本武士, 小笠原秀人 : SPI 活動の可視化 ～フィードバックを重視した計測プロセス構築の全社的な取り組み～, ソフトウェア・シンポジウム (SS2005), 2005.

16. 小笠原秀人, 青木裕伸: ワークショップをベースにした全員参加によるプロセス改善活動, ソフトウェア・シンポジウム (SS2006), 2006.

17. 小笠原秀人, 青木裕伸, 鷲見毅: ワークショップをベースにしたプロセス改善活活動 ～問題解決を出発点とするために～,SPI Japan 2006, 2006.

18. 小笠原秀人: 企業における効果的なトレーニングの実践方法を探る, ソフトウェア・シンポジウム (SS2007), 2007.

19. 羽原寿和, 小笠原秀人: バグ管理ツールの普及とノウハウの蓄積を目指した推進活動事例, 第 26 回ソフトウェア品質シンポジウム (SQiP2007), 2007.

20. 小笠原秀人: テスト管理の課題とその解決に向けた取り組み, 第 28 回ソフトウェア品質シンポジウム (SQiP2009), 2009.

21. 久連石圭, 河村透, 小笠原秀人, 佐々木愛美: テスト管理システムを用いたテスト状況の可視化方法の提案, 情報処理学会第 73 回全国大会, 2011.

22. 小笠原秀人, 河村透, 久連石圭, 佐々木愛美: 導入容易なテスト管理システムの開発とその適用効果, 第 31 回ソフトウェア・シンポジウム (SS201), 2011.

23. 鷲見毅, 加瀬直樹, 市田憲明, 小笠原秀人: テスト設計手法 PROST!, 第 10 回情報科学技術フォーラム (FIT2011), 2011.

24. 中野隆司, 小笠原秀人, 松本智司: 組み合わせテスト技術の導入・定着への取り組み, および上流設計への適用検討の事例, ソフトウェア品質シンポジウム (SQiP2011), 2011.

25. 小笠原秀人, 艸薙匠, 会沢実: 全社的なソフトウェアプロセス改善活動の実践結果とその振り返り, ソフトウェア品質シンポジウム (SQiP2011), 2011.

招待講演

1. Hideto Ogasawara, Takumi Kusanagi,Mikako Arami, Takeshi Tanaka,Minoru Aizawa, Atsushi Yamada : Process Improvement Activities Using CMM SEA1999(Australia), 1999(April).

2. Hideto Ogasawara: Evolution of Software Process Improvement Activities in a Large-scale Organization, KOREA SEPG Conference 2004.

3. 小笠原秀人：大規模組織における定着を重視したプロセス改善活動, 第 4 回クリティカルソフトウェアワークショップ（WOCS2005),2005.

4. 野中誠, 小笠原秀人：定量的管理を活かす経営ビジョン・仕掛け・人材育成, 第 27 回ソフトウェア品質シンポジウム (SQiP2008) ,2008.

5. 小笠原秀人：10 年目の SPI ～推進結果と今後の展望～, 第 30 回ソフトウェア・シンポジウム（SS2010）, 2010.

6. 小笠原秀人 : SPI の動向と発表事例の概要紹介, 第 8 回クリティカルソフトウェアワークショップ,2011.

共著書

1. 松本吉宏編, ソフトウェア現場力ハンドブック, オーム社,2009.（第 1 編第 6 章 6.4 品質保証の 6.4.3 レビュー技法,6.4.4 テスト技法を担当）

共訳書

1. 飯塚悦功, 西康晴 監訳, 株式会社東芝 S&S 研究所 訳,Roger S.Pressman 著：実践ソフトウェア工学, 日科技連出版,2000.（27 章 リエンジニアリング,28 章 クライアント・サーバソフトウェア工学を担当）

解説論文

1. 小笠原秀人, 小島昌一：定着を重視したプロセス改善活動, 情報処理学会誌,2003 .

2. 小笠原秀人：高度化する不具合管理への適応 ～不具合管理システムを利用した品質向上活動～, クオリティマネジメント誌,2010.

雑誌記事

1. 小笠原秀人, 田中武志：品質評価・管理のための手法とツールの実践的運用, 東芝レビュー Vol.51 No.2,1996.

2. 小笠原秀人, 會沢実, 山田淳：イントラネットを利用した分散開発環境下でのソフトウェア品質管理, 東芝レビュー Vol.53 No.8,1998.

3. 小笠原秀人, 中野一男, 田中史朗：ソフトウェア開発プロセスの評価と改善, 東芝レビュー Vol.56 No.11,2001.

4. 小笠原秀人：はじめようレビュー活動！,ThinkIT,2009/3/6,3/13,3/27
   第 1 回：はじめる前に頭に入れておきたいこと！（http://thinkit.jp/article/874/1/）
   第 2 回：本当に役立つレビュー教育を行うには（http://thinkit.jp/article/892/1/）
   第 3 回：刻み込め！ピアレビューの心得（http://thinkit.jp/article/902/1/）

5. 小笠原秀人, 中野隆司, 大谷和夫：組合せ技術を利用したソフトウェアテスト, 東芝レビュー Vol.64 No.4,2009.

6. 河村透, 小笠原秀人:ソフトウェアのテスト管理システム, 東芝レビュー Vol 66 No.1,2011.

7. 小笠原秀人:SQuBOK 中国に渡る, ソフトウェア品質のホンネ, 2011/2/15,2/23,3/1,3/9
   第 1 回：SQuBOK 中国に渡る その 1
   （http://www.juse-sqip.jp/honne/backnumber_004.html）
   第 2 回：SQuBOK 中国に渡る その 2
   （http://www.juse-sqip.jp/honne/backnumber_005.html）

第 3 回：SQuBOK 中国に渡る その 3
（http://www.juse-sqip.jp/honne/backnumber_006.html）
第 4 回：SQuBOK 中国に渡る その 4
（http://www.juse-sqip.jp/honne/backnumber_007.html）

関連論文の印刷公表の方法及び時期

全著者名： Hideto Ogasawara, Atsushi Yamada, Takumi Kusanagi, Mikako Arami

論文題目： How to Effectively Promote the Software Process Improvement Activities in a Large-Scale Organization

印刷公表の方法及び時期： 平成 14 年 6 月　Proc. of Software Quality - ECSQ 2002, 7th International Conference, Springer, pp.124-134, 2002
（本論文の第 3 章と関連）

全著者名： Hideto Ogasawara , Noboru Fujimaki, Shoichi Kojima

論文題目： The Evolution of Software Process Improvement Activities in a Large-Scale Organization

印刷公表の方法及び時期： 平成 16 年 9 月　Proc. of CONference on QUality Engineering in Software Technology, 2004
（本論文の第 3 章と関連）

全著者名： Hideto Ogasawara, Takumi Kusanagi, Noboru Fujimaki

論文題目： Practical Approach to Evolve SPI Activities in a Large-Scale Organization

印刷公表の方法及び時期： 平成 17 年 9 月　Proc. of 3rd World Congress for Software Quality, pp.I-181-I-190, 2005
（本論文の第 3 章と関連）

全著者名： 小笠原 秀人, 藤巻 昇, 艸薙 匠, 田原 康之, 大須賀 昭彦

論文題目： 大規模組織におけるソフトウェアプロセス改善活動の適用評価 — 10 年間の実践に基づく考察

印刷公表の方法及び時期： 平成 22 年 9 月　情報処理学会論文誌, Vol.36, No.05, pp.1070-1080, 2010
（本論文の第 4 章と関連）

全著者名： Hideto Ogasawara, Takashi Ishikawa, Tetsuro Moriya

論文題目： Practical approach to development of SPI activities in a large organization : Toshiba's SPI history since 2000

印刷公表の方法及び時期： 平成 18 年 6 月　Proc. of 28th International Conference on Software Engineering, pp.595-599, 2006
（本論文の第 4 章と関連）

全著者名： Hideto Ogasawara, Atsushi Yamada, Michiko Kojo

論文題目： Experiences of Software Quality Management Using Metrics through the Life-Cycle

印刷公表の方法及び時期： 平成 8 年 3 月　Proc. of 18th International Conference on Software Engineering, pp.179-188, 1996
（本論文の付録 A と関連）

全著者名： Hideto Ogasawara, Minoru Aizawa, Atsushi Yamada

論文題目： Experiences with Program Static Analysis

印刷公表の方法及び時期： 平成 10 年 11 月　Proc. of Fifth International Software Metrics Symposium, pp.109-112, 1998
（本論文の付録 B と関連）

全著者名： Hideto Ogasawara, Takumi Kusanagi, Mikako Arami

論文題目： Process Improvement Activities by High Quality Software Creation Support Virtual Center

印刷公表の方法及び時期： 平成 12 年 9 月　Proc. of 2rd World Congress for Software Quality, pp.769-774, 2000
（本論文の付録 B と関連）

著者略歴

小笠原 秀人（おがさわら ひでと）

昭和 63 年 3 月 31 日	東京理科大学 工学部 第 2 部 経営工学科 卒業
平成 2 年 3 月 31 日	東京理科大学 大学院 工学研究科 経営工学専攻 修士課程 修了
平成 2 年 4 月 1 日	株式会社東芝 入社
平成 21 年 4 月 1 日	電気通信大学 大学院 情報システム学研究科 社会知能情報学専攻 博士課程 入学
平成 23 年 12 月 31 日	電気通信大学 大学院 情報システム学研究科 社会知能情報学専攻 博士課程 修了予定