NAIST-IS-DD0761024

博士論文

ソフトウェア開発における
定量的プロセス管理の実施支援に関する研究

伏田 享平

2010年2月4日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

伏田 享平

審査委員：

飯田 元 教授	（主指導教員）
西谷 紘一 教授	（副指導教員）
松本 健一 教授	（副指導教員）
楠本 真二 教授	（大阪大学）
門田 暁人 准教授	

ソフトウェア開発における
定量的プロセス管理の実施支援に関する研究*

伏田 享平

内容梗概

近年，ソフトウェアの果たす社会的役割は重大になっており，ソフトウェアを高品質に生産するための技術が重要となっている．そのためには，ソフトウェアの開発プロセス（ソフトウェア開発に関わる作業の系列）の実態を定量的に計測，把握し，問題が生じた場合に対策を行う，いわゆる「定量的プロセス管理」の実施が強く求められている．定量的プロセス管理を実施することで，開発プロセスの評価から属人性を排除し，自動的に評価を行うことができる．しかし，実際には，定量的プロセス管理を実施しているソフトウェア開発組織の数は少ない．また，定量的プロセス管理を実施している組織であっても，プロジェクトが成功しているのは，定量的プロセス管理を実施している組織の 40% 程度であるという調査結果もある．すなわち，定量的プロセス管理の実践にあたっては，何らかの障害が存在していることが考えられる．

本論文では，ソフトウェア開発現場において，定量データを用いたプロセス管理がどのように実施されているかを確認するため，ある国内のソフトウェア開発組織を対象に定量データの利用調査を行った．調査の結果，組織標準として用意されている定量データの多くはプロジェクト管理に有効に活用されている一方で，組織において収集体制が整備されていないものや，意思決定を行う際の判断基準となる基準値が設定されていないものなど，ある属性を持つデータは多くのプロジェクトでは利用されていないことが明らかになった．また，プロジェクトの実態にあわせて，個々のプロジェクト管理者によって収集するデータの個数や種類などの調整を行っていることを確認した．この調査の結果，実際に定量的プロセス管理を行うにあたって

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文，NAIST-IS-DD0761024, 2010 年 2 月 4 日.

は，(A) 定量データの測定計画立案に関する課題，(B) 開発プロセスの定量的評価・分析に関する課題の 2 種類の課題が存在することを確認した．

次に，調査の結果を受け，定量データを用いた開発プロセスの管理に関して，その管理計画の作成を支援する枠組みを構築した．提案する枠組みでは，プロセス管理に用いる定量データの測定・分析活動をプロジェクトの状況にあわせて調整する際の，体系的な枠組みを提供した．また，この枠組みのもとで，開発プロセス，およびその管理プロセスを容易に策定できるよう支援するシステム AQUAMarine を開発した．ソフトウェア開発組織のプロジェクト管理者に AQUAMarine のレビューを実施してもらい，AQUAMarine がプロジェクト管理計画立案作業に対して有効な支援を行うことができることを確認した．

最後に，ソフトウェアの開発プロセスにおいて，ソフトウェアの品質に影響する作業がどのように行われたかを明らかにする手法を提案する．提案手法では，ソフトウェアの開発中に自動的に収集された作業記録を利用し，開発プロセスの分析手法および評価尺度を定める．提案する評価尺度を用いてプロダクトの品質とプロセスの品質との相関を算出した結果，プロダクトの品質とプロセスの品質との間に関係があることを確認した．また，提案手法を実際の開発プロジェクトデータに適用した結果，開発プロセス中の，ソフトウェアの設計品質を向上させる作業が行われている箇所を特定することが可能であることを確認した．

キーワード

ソフトウェアプロセス，定量的管理，計画立案支援，リポジトリマイニング，プロセス評価，デザインパターン

Methods for Analysis and Control of Software Development Process Based on Quantitative Data*

Kyohei Fushida

Abstract

Quantitative management is a key technology in software process management and improvement. In order to develop software with high quality, we should assess not only quality of the software but also the quality of its development processes. Process assessment is important as it enables identification of problems in the early development stages.

In order to ensure objectivity and assess automatically, process assessment should be quantitative. However, there are two problems about quantitative process management. First, previous study have not sufficiently investigated the effectiveness and the methodology of planning quantitative management. Secondly, previous study have not sufficiently investigated the methodology of analyzing and assessing software development process quantitatively.

In this dissertation, we study the status of quantitative management in a software organization. Our results show that a lot of indicators are adopted though some are unused or customized according to circumstances. Based on the results, we show the policy to use indicators for quantitative management in a software organization.

Furthermore, based on the result, we proposed a framework for authoring and

*Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0761024, February 4, 2010.

tailoring software development processes with quantitative management. The proposed framework systematically organizes organization/project-level procedures for planning quantitative management. Also we introduced the AQUAMarine system that supports the project process planner to plan/perform/tailor development processes with quantitative management. The AQUAMarine system has a feature to embed the activities of collecting or analyzing quantitative data into development processes. The AQUAMarine system was evaluated by a software organization. The results show that the system can assist in planning quantitative management appropriately.

Finally, we propose a method and metrics to analyze process quality using empirical data. Empirical data can be collected automatically using software management tools. We apply our method to open source software projects. As a result, we find relationships between some metrics of bug fixing processes and product quality. In addition, we can specify the work for improving software design in development process.

Keywords:

Software Process, Quantitative Management, Planning Support, Repository Mining, Process Assessment, Design Pattern

目次

図目次

表目次

関連発表論文

学術論文誌

1. 伏田享平, 高田純, 米光哲哉, 福地豊, 川口真司, 飯田元. AQUAMarine: 定量的管理計画立案システム. *SEC journal*, Vol. 5, No. 4, pp. 244–251, September 2009. （第 4 章，第 5 章に関連する）
2. 伏田享平, 米光哲哉, 福地豊, 川口真司, 飯田元. 定量的管理指標の利用実態調査〜あるソフトウェア開発組織を対象として〜. Vol. 51, No. 5, 情報処理学会論文誌. （採録決定）（第 3 章に関連する）

査読付き国際会議・ワークショップ

1. Kyohei Fushida, Shinji Kawaguchi, and Hajimu Iida. A method to investigate software evolutions using design pattern detection tool. In *Proceedings of the 1st International Workshop on Software Patterns and Quality (SPAQu'07)*, pp. 11–16, December 2007. Nagoya, Japan. （第 7 章に関連する）
2. Kyohei Fushida, Jun Takata, Masataka Nagura, Shinji Kawaguchi, and Hajimu Iida. VSOP framework: A framework for validating suppliers' processes. In *Proceedings of the 9th International Conference on Product Focused Software Process Improvement (PROFES 2008) (Short Paper Session)*, pp. 9–12, June 2008. Frascati, Italy. （第 5 章に関連する）
3. Kyohei Fushida, Jun Takata, Tetsuya Yonemitsu, Yutaka Fukuchi, Shinji

Kawaguchi, and Hajimu Iida. A framework and system for planning and tailoring software development processes with quantitative management. In *Proceedings of the 4th International Conference on Project Management (ProMAC 2008)*, pp. 286–294, September 2008. Anchorage, Alaska USA. （第 3 章，第 4 章，第 5 章に関連する）

査読付き国内研究集会

1. 伏田享平, 亀井靖高, 川口真司, 飯田元. 定量的測定データの体系化に基づいた開発プロセステーラリング方式の提案. 満田成紀, 羽生田栄一（編）, ソフトウェアエンジニアリング最前線 2006，ソフトウェアエンジニアリングシンポジウム 2006（SES2006）, pp. 51–58, October 2006. （第 4 章，第 5 章に関連する）
2. 伏田享平. 企業間の請負開発プロセス検証を目的とした枠組みの提案. 阿萬裕久（編）, ウィンターワークショップ 2008・イン・道後 論文集, pp. 107–108, January 2008. （第 5 章に関連する）
3. 伏田享平, 名倉正剛, 川口真司, 飯田元. 適用履歴に着目したデザインパターンメトリクスの提案. ソフトウェアエンジニアリングシンポジウム 2008 ワークショップ論文集, pp. 29–30, September 2008. （第 7 章に関連する）
4. 伏田享平, 川口真司, 飯田元. プロセスメトリクスを用いた開発プロセス評価手法の確立に向けて. ソフトウェアエンジニアリングシンポジウム 2009 ワークショップ「ソフトウェア開発マネジメントのための測定と分析」, September 2009. （第 6 章に関連する）

査読無し国内研究集会

1. 伏田享平, 川口真司, 飯田元. 定量的管理を取り入れたソフトウェア開発計画立案支援システムの提案. ソフトウェア信頼性研究会 第 3 回ワークショップ論文集, pp. 27–36, July 2006. （第 4 章，第 5 章に関連する）
2. 伏田享平, 大前勇輝, 名倉正剛, 川口真司, 大蔵君治, 飯田元. バグ管理システ

ムを対象としたアジャイルソフトウェア開発における保守プロセスの観察. 電子情報通信学会技術報告, ソフトウェアサイエンス研究会, 第 SS2008-39 巻, pp. 1–6, November 2008. （第 6 章に関連する）

3. 伏田享平, 川口真司, 飯田元. バグ管理システムのログを利用した保守プロセス評価メトリクス. 電子情報通信学会技術研究報告, 第 109 巻, pp. 89–94, August 2009. （第 6 章に関連する）

その他の発表論文

査読付き国際会議・ワークショップ

1. Kazumasa Hikichi, Kyohei Fushida, Hajimu Iida, and Ken'ichi Matsumoto. A software process tailoring system focusing to quantitative management plans. In Jürgen Münch and Matias Vierimaa, editors, *Proceedings of the 7th International Conference on Product Focused Software Process Improvement (Profes2006)*, Vol. 4034, pp. 441 – 446, June 2006. Amsterdam, Netherlands.
2. Kazumasa Hikichi, Tetsuya Yonemitsu, Yutaka Fukuchi, Kyohei Fushida, and Hajimu Iida. An assistance method of incorporating quantitative management indicator into software development process. In *Proceedings of the 3rd International Conference on Project Management(ProMAC2006)*, September 2006. Sydney, Australia.
3. Shuji Morisaki, Tomoko Matsumura, Kimiharu Ohkura, Kyohei Fushida, Shinji Kawaguchi, and Hajimu Iida. Fine-grained software process analysis to ongoing distributed software development. In *Proceedings of the 1st Workshop on Measurement-based Cockpits for Distributed Software and Systems Engineering Projects (SOFTPIT 2007)*, pp. 26–30, August 2007. Munich, Germany.
4. Masateru Tsunoda, Kohei Mitsui, Kyohei Fushida, Yasutaka Kamei, Masahide Nakamura, Keita Goto, and Ken'ichi Matsumoto. An authentication method combining spatiotemporal information and actions. In

Proceedings of the 4th International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2008), pp. 41–49, June 2008. Tokyo, Japan.

5. Yoshihiko Fukushima, Raula Kula, Shinji Kawaguchi, Kyohei Fushida, Masataka Nagura, and Hajimu Iida. Code clone graph metrics for detecting diffused code clones. In *Proceedings of the 16th Asia-Pacific Software Engineering Conference (APSEC 2009)*, pp. 373–380, December 2009. Penang, Malaysia.

査読付き国内研究集会

1. 高田純, 伏田享平. 定量的管理システムの実施を支援するシステムの提案. ソフトウェアエンジニアリングシンポジウム 2008 ワークショップ論文集, pp. 17–18, September 2008.
2. 福島義彦, 川口真司, 伏田享平, 名倉正剛, 飯田元. コードクローンが存在するファイルに基づくコードクローン可視化手法. 鵜林尚靖, 岸知二（編）, ソフトウェアエンジニアリング最前線 2009，ソフトウェアエンジニアリングシンポジウム 2009（SES2009）, pp. 105–110, September 2009.

査読無し国内研究集会

1. 森崎修司, 松村知子, 大蔵君治, 伏田享平, 川口真司, 飯田元. エンピリカルデータを対象としたマイクロプロセス分析. 情報処理学会研究報告, 第 2006 巻, pp. 9–15, November 2006.
2. 角田雅照, 伏田享平, 三井康平, 亀井靖高, 後藤慶多, 中村匡秀, 松本健一. 位置と速度を利用した移動体向け認証方式の提案. 電子情報通信学会技術報告, 第 MoMuC2006-55 巻, pp. 11–16, November 2006.
3. 高田純, 伏田享平, 米光哲哉, 福地豊, 川口真司, 飯田元. 定量的管理計画支援環境のための WBS オーサリングツール群の開発. 日本ソフトウェア科学会第 24 回大会論文集 CD-ROM(講演番号 7B-3), September 2007.

4. 山科隆伸, 上野秀剛, 伏田享平, 亀井靖高, 名倉正剛, 川口真司, 飯田元. レガシーソフトウェア保守プロセスにおける開発者によるコードクローン認識についての観察. 情報処理学会第 70 回全国大会, 第 5 巻, pp. 411–412, March 2008.
5. 高田純, 伏田享平, 名倉正剛, 川口真司, 飯田元. ソフトウェア開発プロセスにおける定量的管理計画の立案・共有支援システム. 情報処理学会第 70 回全国大会, 第 5 巻, pp. 413–414, March 2008.
6. 山科隆伸, 上野秀剛, 伏田享平, 亀井靖高, 名倉正剛, 川口真司, 飯田元. コードクローンに着目したソフトウェア保守支援ツールの設計と実装. 電子情報通信学会技術研究報告, 第 108 巻, pp. 65–70, May 2008.
7. 石田響子, 伏田享平, 名倉正剛, 川口真司, 飯田元. 「非障害案件」の発見を目的としたバグ票の観察. 情報処理学会研究報告, 第 2008 巻, pp. 1–8, September 2008.
8. 水野恵祐, 伏田享平, 川口真司, 飯田元. 複数手法の併用によるデザインパターン検出結果の改善. 電子情報通信学会技術研究報告, 第 109 巻, pp. 39–44, September 2009.
9. 奥村哲也, 大蔵君治, 伏田享平, 川口真司, 名倉正剛, 飯田元. ソフトウェアタグの適用を支援する開発履歴可視化ツール. 情報処理学会研究報告, 第 2009-SE-166 巻, pp. 1–8, October 2009.
10. 片山真一, 大蔵君治, 伏田享平, 川口真司, 名倉正剛, 門田暁人, 飯田元. ソフトウェアタグを用いた設計文書メトリクスからの低品質モジュールの予測. 電子情報通信学会技術研究報告, 第 109 巻, pp. 67–72, December 2009.

紀要・テクニカルレポート

1. Takanobu Yamashina, Hidetake Uwano, Kyohei Fushida, Yasutaka Kamei, Masataka Nagura, Shinji Kawaguchi, and Hajimu Iida. SHINOBI: A real-time code clone detection tool for software maintenance. NAIST Information Science Technical Report NAIST-IS-TR2008005, Nara Institute of Science and Technology, March 2008.

2. 高田純, 伏田享平, 川口真司, 飯田元. 定量的管理計画の立案を支援するシステムの開発. NAIST Information Science Technical Report NAIST-IS-TR2008008, Nara Institute of Science and Technology, November 2008.

第 1 章

序論

1.1. 研究の背景・目的

近年，社会におけるソフトウェアの役割の重要性が増大している．一方で，ソフトウェア開発プロジェクトがソフトウェアのリリースに至らず失敗に終わることや，欠陥が見過ごされたままリリースされてしまい，莫大な社会的損失を生み出している事例が報告されている．例えば，米国では，ソフトウェアの欠陥による年間損失額は 600 億ドルにのぼる [72]．また，日本においても，証券システムや銀行の ATM システム，自動改札機システムの停止等，ソフトウェアの欠陥が原因となった大規模な障害が発生している．この原因として，ソフトウェア開発プロジェクトの規模が年々大きくなっているにも関わらず短期で開発することが求められ，開発計画を熟慮しないうちに開発が進行することが挙げられる．その結果，計画通りに開発プロジェクトが進行せず，スケジュールの遅れやコスト超過などの問題が生じることが多くなっている．

ソフトウェアを高品質に生産するためには，ソフトウェア開発中にソフトウェア開発プロセス（ソフトウェア開発に関わる作業の系列）を評価するプロセス評価を行うことが重要である．プロセス評価はソフトウェアの開発中に随時実施でき，早期に問題を検出することが可能である．このため近年では特に注目されており，開発組織においてプロセスを改善するための枠組みが多く提案されている [13, 52]．

多くの場合，開発プロセスの改善は，図 1.1 のような PDCA サイクルに従って実施される．開発を実施する際には，従来の実績や将来の予測をもとに計画を立てる．

開発実施後は，計画に沿っていたかを確認し，計画に沿っていなかった部分を分析し原因を追及する．この結果をもとに，次回のプロジェクトでは開発プロセスを改善する．開発プロセスの改善にあたっては，図 1.1 に示すように，計画，実行，評価，改善，に関する活動を順に実施する．そして，改善の結果を次回実施するプロジェクトの計画に反映させ，再び図 1.1 のサイクルを回す．このように継続的に改善を続けていくことで，プロセスの品質を向上させていく．

このような開発プロセスの改善にあたっては，定量的に開発プロセスの管理を行った上で，改善を行うことが望ましい．定量的プロセス管理（以降，定量的管理と略す）とは，定量的測定データ（以下，定量データと略す）を開発時に収集し，そこから導かれる評価指標（管理指標）に基づいて管理する手法である．定量的管理を行う場合には，プロジェクトごとに管理指標の取捨選択や測定する定量データの調整（カスタマイゼーション）を行う [81]．定量的にプロセス管理を行うことで，分析する際の属人性が排除でき，客観的な評価のもとで，開発プロセスを管理し，改善につなげていくことが可能となる．

しかし，ソフトウェア開発における定量的管理において，以下のような課題が存在

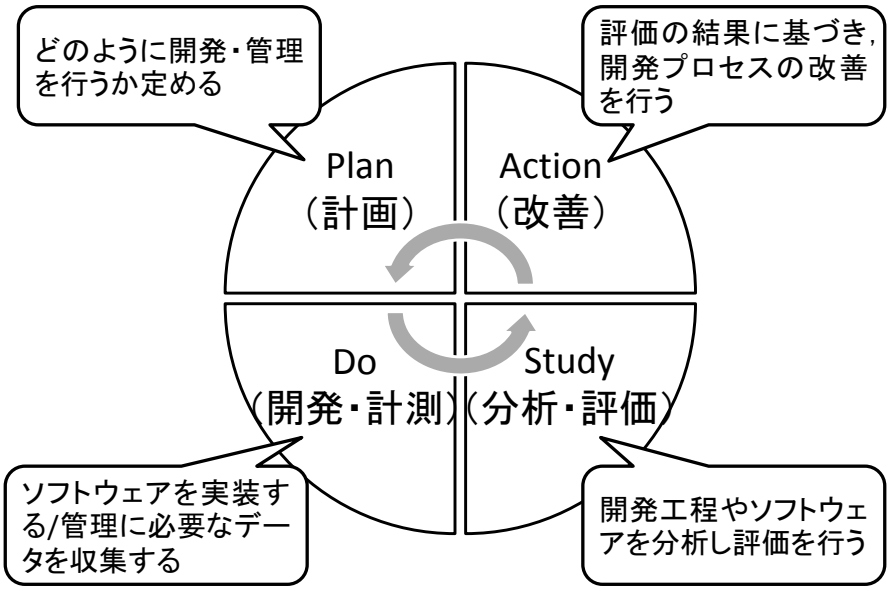

図 1.1　ソフトウェア開発における PDCA サイクルと本研究で扱う課題との関連

する．

(課題) ソフトウェア開発における定量的管理実施に関する課題

実際のソフトウェア開発組織において定量的管理がどのように実施されているかが明らかになっていない．定量的なプロセス管理を行い，ソフトウェア開発プロジェクトを成功に導くためには，実態を明らかにした上で，何らかの障害が発生している場合には，しかるべき対策を講じる必要がある．

本論文では，定量的管理がどのように行われているかを確認するため，まず日本国内のあるソフトウェア開発組織において，定量的管理がどのように実施されているのかを調査した．調査では，開発プロジェクトの管理者（プロジェクトマネージャ）が，個々のプロジェクトにおいてどのように定量的管理を実施しているかヒアリングを行った．その結果，ソフトウェア開発における定量的管理実施に関する課題として，次の 2 つの課題が存在することが明らかになった．

(課題 A) 定量データの測定計画立案に関する課題

定量的管理を行うためには，開発計画を立案する際にどのような種類の定量データをどのタイミングで収集すべきかを考慮する必要がある．しかし，一般に 1 つのプロジェクトから収集することができる定量データの候補は膨大であり，個々のプロジェクトにおいて，どのような目的に沿って，どのような定量データを収集する必要があるかを把握することは難しい．また，プロジェクトによって，収集する定量データやその収集・分析方法は異なることが予想されるが，収集するデータやその分析方法等の決定は，個々のプロジェクト管理者の経験に大きく依存している．

(課題 B) 開発プロセスの定量的評価・分析に関する課題

ソースコードの変更履歴や不具合の報告記録などの開発作業記録を自動的に収集する研究は多数行われており，多くの開発組織ではこのような開発作業記録を自動収集する仕組みが導入されている．しかし，定量的に開発プロセスを評価するための基準やその方法論は確立されておらず，どのような観点からプロセスを分析，評価すれば良いかが明らかになっていない．

本論文では，ソフトウェア開発において定量的管理を行う上で障害となっている，これら課題 A および課題 B の解決法について検討を行った．まず，課題 A に対して，ソフトウェア開発プロジェクトにおける定量的管理計画立案のための枠組みを構築した．この枠組みでは，測定方法を定量データの「粒度」と「測定頻度」の 2 つの観点から分類し，プロジェクトの状況に合わせた調整方法を提供する．また，このフレームワークのもとで，定量的管理計画の立案作業を支援するためのシステムを開発した．

次に，課題 B に対して，ソフトウェア開発作業実施時に自動的に取得できるデータを用いた定量的な開発プロセス分析手法を提案した．現在，多くの開発プロジェクトでは，ソフトウェア開発作業に関するデータを，ソフトウェアリポジトリと呼ばれるデータストアに自動的に格納している．提案手法は，このような時系列に自動的に収集された開発作業記録から，個々の開発作業がどのように実施されたかを抽出する．そして，抽出した作業手順から，ソースコードの変更回数や作業に関わった開発要員の数，作業の手戻り回数など，実施された作業手順を特徴づける定量的なデータを機械的に抽出する．この定量データを用いて，実施された作業（開発プロセス）の分析を行う．これにより，提案手法によってその特徴を表すデータを用いて開発プロセスを定量的に評価，分析することが可能となった．また，自動収集された開発記録を利用しており，人手をかけず従来よりも細かい視点で開発プロセスの分析を行うことが可能となった．さらに，ソフトウェアに適用されたデザインパターンと呼ばれるソフトウェア設計におけるノウハウに着目し，デザインパターンの適用状況を時系列に観察することでソフトウェア設計がどのように変更されていったかを分析する手法を提案した．

1.2. 本論文の構成

本論文の構成は以下の通りである．第 2 章では，定量データを用いたプロジェクト管理について概説し，課題を整理する．第 3 章では，国内のソフトウェア開発組織を対象に行った，プロジェクト管理における定量データの利用実態調査の結果を示し，そこから導かれる課題について議論する．第 4 章では，第 3 章での議論をうけて，定量データ測定計画立案のための枠組みについて述べる．続く第 5 章では第 4

章で提案した枠組みに沿って，計画立案を支援するシステム AQUAMarine の開発について述べる．第 6 章では，自動的に収集された定量データを用いた開発プロセスの定量的評価・分析手法について述べる．第 7 章では，ソフトウェアに適用されたデザインパターンに着目した，ソフトウェアの進化プロセス分析手法について述べる．最後に第 8 章で全体のまとめを述べ，本論文を結ぶ．

第 2 章

ソフトウェア開発における定量的プロセス管理

本章では，定量的管理の一般的な流れについて説明する．また，ソフトウェア開発プロセス，およびその定量的管理に関する関連研究，規格について概説する．具体的には，開発プロセスの計画，計測，分析・評価に関する研究について述べる．最後に，これらの関連研究をふまえて，ソフトウェア開発における定量的管理の課題について整理する．

2.1. 定量的管理の一般的な流れ

一般に，ソフトウェア開発プロジェクトにおける定量的管理は，標準開発プロセス定義（各工程で発生する作業の内容を，組織内のどのプロジェクトでも利用できるように一般化した形で記述したもの）を策定する**プロセスエンジニア**，開発・管理計画の立案を行う**プロジェクト計画者**，計画者の立案した計画に従い管理に必要なデータの測定を行う**測定者**，測定されたデータをもとに分析・開発プロセスへのフィードバックを行う**分析者**，の 4 つの役割によって実施される．図 2.1 に定量的管理の流れを示す．一般的に，定量的管理を取り入れたプロジェクト管理は，以下の手順で行われる．

1. プロセスエンジニアによる標準開発プロセス定義の策定

プロセスエンジニアは，組織内のプロセス構築・改善を目的として，組織の特性や実状を考慮し，標準開発プロセス定義を作成する．また，プロジェクトの実行中に得られた知見を用いて，標準開発プロセス定義を継続的に修正する．さらに，個々のプロジェクト開発管理計画を立案する際には，プロジェクト計画者と協力し，標準開発プロセス定義をプロジェクトごとに適宜修正して組織内に展開する．

2. プロジェクト計画者によるプロジェクト開発管理計画の立案

定量的管理を取り入れたプロジェクトを立ち上げる際には，まず初めにプロジェクト計画者がプロジェクト開発管理計画を立案する．プロジェクト計画者は，開発組織で定められた組織標準プロセスモデルや CMMI などのフレームワークをもとに，プロジェクトの予算や人員，開発規模を考慮した上で，最適な管理計画を作成し，管理に必要となるデータの測定と分析活動を定め，その手順を確立する．

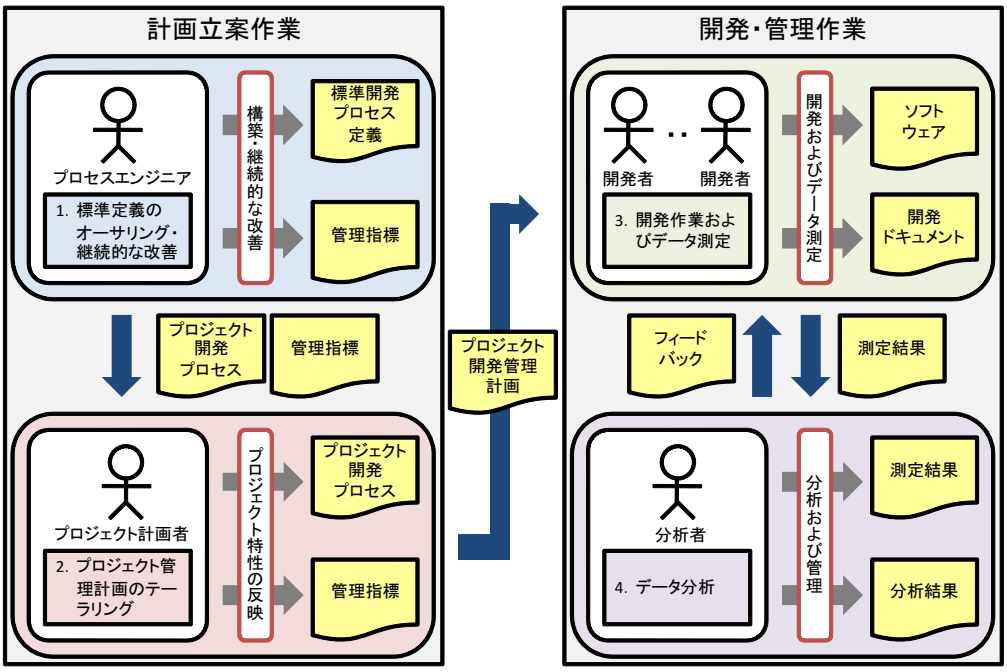

図 2.1　一般的な定量的管理の流れ

3. 測定者によるデータの測定

測定者は計画者によって作成されたプロセス計画に従って，必要なデータの測定・収集を行い，分析に適した形に整理して分析者に報告する．測定者は，一般的な開発組織では開発者が兼ねることが多い．ここでのデータ測定は，測定者個人の進捗を測定するものではなく，開発プロジェクト全体での品質・進捗の度合いを推定するために行う．そのため，測定者には測定の目的，方法などを十分理解して，客観的な値を正確に測定することが求められる．

4. 分析者によるデータの分析・プロジェクト管理

分析者は，測定者から報告されたデータを分析し，管理図表（グラフ，表など）を用いて，定められた判断基準（目標値，標準値）と比較することにより，品質・進捗の管理を行う．個々のデータは，そのままでは品質・進捗について直接的に把握することができない値であるため，グラフ化などを行って分析を行う必要がある．そのため，分析者は，データの分析方法や，管理図表の見方を十分に理解する必要がある．また定められた判断基準と測定された実績値との比較を行い，もし目標値に対して実績値が著しく逸脱した場合，もしくは放置すれば逸脱する傾向やリスクがあるとわかった場合，分析者は直ちに適切な是正措置をとる必要がある．

2.2. 定量的管理に関連する研究

定量データを用いたソフトウェア開発の進捗管理や品質管理の改善・効率化は，生産性や品質の向上に直接的に影響する要素であり，これまでにも盛んに研究が行われている．Basili らは，プロジェクトの管理目的を定め，次に定めた管理目的の達成度を測るための定量的指標との関連付けを行い，実践を行うためのパラダイムである GQM (Goal - Question - Metrics) を提唱している [9]．GQM と類似の概念としては，McGarry らにより提案され，ISO/IEC 15939[35] として規格化されたソフトウェア計測プロセスである PSM (Practical Software Measurement) [52] がある．また，ISO/IEC 15504 規格 [36, 39–41, 43] では，ソフトウェアプロセスのアセスメントモデルが定義され，開発組織のプロセス能力を評価するための様々な測定項目が

定められている．他にも，CMM や CMMI，IDEAL[25] などの参照モデルがある．

プロセス能力を測定するための規格としては, ISO/IEC 15504 規格 [36, 39–41, 43] が挙げられる．ISO/IEC 15504 規格では，ソフトウェアプロセスの参照モデルが定義され，プロセス能力を評価する上での様々な測定項目が定められている．ソフトウェアそのものの品質測定に関する規格として，ISO/IEC 9126 規格 [34, 37, 38, 42] によるソフトウェア品質特性の定義がある．

以下では，本研究と特に関連の深い規格である ISO/IEC 12207[33] 規格，および ISO/IEC 15939 規格，CMMI について述べる．また，定量的管理の実施を支援する，開発プロジェクトの計画立案支援，定量データの収集支援，ソフトウェアの品質評価に関する研究についてもあわせて述べる．

2.2.1. 関連規格

ISO/IEC 12207 規格

ISO/IEC 12207（邦訳：JIS X0160[61]）は，ソフトウェアライフサイクルプロセスを定義した国際規格である．ソフトウェアライフサイクルとは，ソフトウェアの構想から作成，廃棄に至るまでの一連のプロセスの流れのことを指す．この規格では，ソフトウェアを作成・管理する際に必要となる共通の枠組みを定義している．この規格で定義されている枠組みでは，ソフトウェアライフサイクルを包容し，ソフトウェア製品およびサービスの取得から供給までのプロセスが定義されている．さらに，この枠組みでは，プロセスの管理および改善についても規定されている．

図 2.2 に，JISX 0160:1996 で定義されているソフトウェアライフサイクルプロセスの構成を示す．この規格では，ソフトウェアライフサイクルにおける活動を 5 つの主ライフサイクルプロセス，8 つの支援ライフサイクルプロセス，4 つの組織に関するライフサイクルプロセスに分類している．加えて，修整プロセスと呼ばれる，これらの定義したプロセスをプロジェクトの特性に合わせ調整するプロセスも定義されている．それぞれのライフサイクルプロセスは，アクティビティに分割され，アクティビティはさらにタスクに分割される．

ISO/IEC 12207 規格に適合するには，修整プロセスに従い，各プロジェクトに応じて，プロセス，アクティビティおよびタスクを取捨選択し，その全てを実行する必要がある．

ただし，ISO/IEC 12207 規格では，作成する文章の名称や様式，その他具体的な内容については指定しない．また，特定のライフサイクルモデルやソフトウェア開発手法については指定していない．そのため，ISO/IEC 12207 規格を適用する場合には，プロジェクトに応じて適切なライフサイクルモデル，開発手法を選択し適用する必要がある．

ISO/IEC 15939 規格

ISO/IEC 15939（邦訳：JIS X0141[62]）は，ソフトウェアの測定プロセスに関する国際規格である．ISO/IEC 15939 規格では，プロジェクト管理や品質保証などの様々な情報ニーズを満たすための測定と分析，解釈のためのフレームワークが示されている．

本論文での定量的管理の概念と用語は，図 2.3 の ISO/IEC 15939 で示されている測定情報についての参照モデルに従う．図 2.3 の測定情報モデルは，具体的な測定結果から組織の意思決定の基礎となる情報成果物がどのような過程で得られるかを階

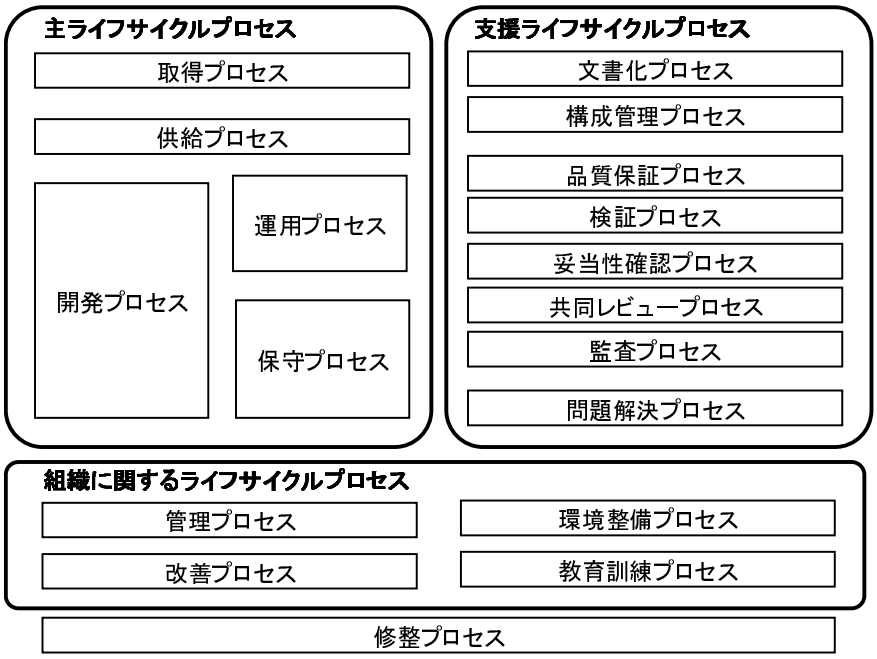

図 2.2　JISX 0160:1996 の構成（JIS X0160:1996 の図 1 をもとに作成）

層構造の形で表したものである．

図 2.3 に示したモデルでは，プロジェクトで実際に測定可能な規模，工数，欠陥数など，プロセスや成果物に固有の属性を意思決定のための指標へ関連付けることで，定量化された情報に基づく客観的な意思決定を容易にする道筋を示している．プロジェクト中に存在する様々な測定可能な属性は，定められた測定方法に従って基本測定量に定量化される．次に，いくつかの基本測定量を，入力として測定関数に与えることで導出測定量が算出される．そして得られた導出測定量を，定められた分析モデルに基づいて分析することで管理指標が得られる．プロジェクトの管理者は，

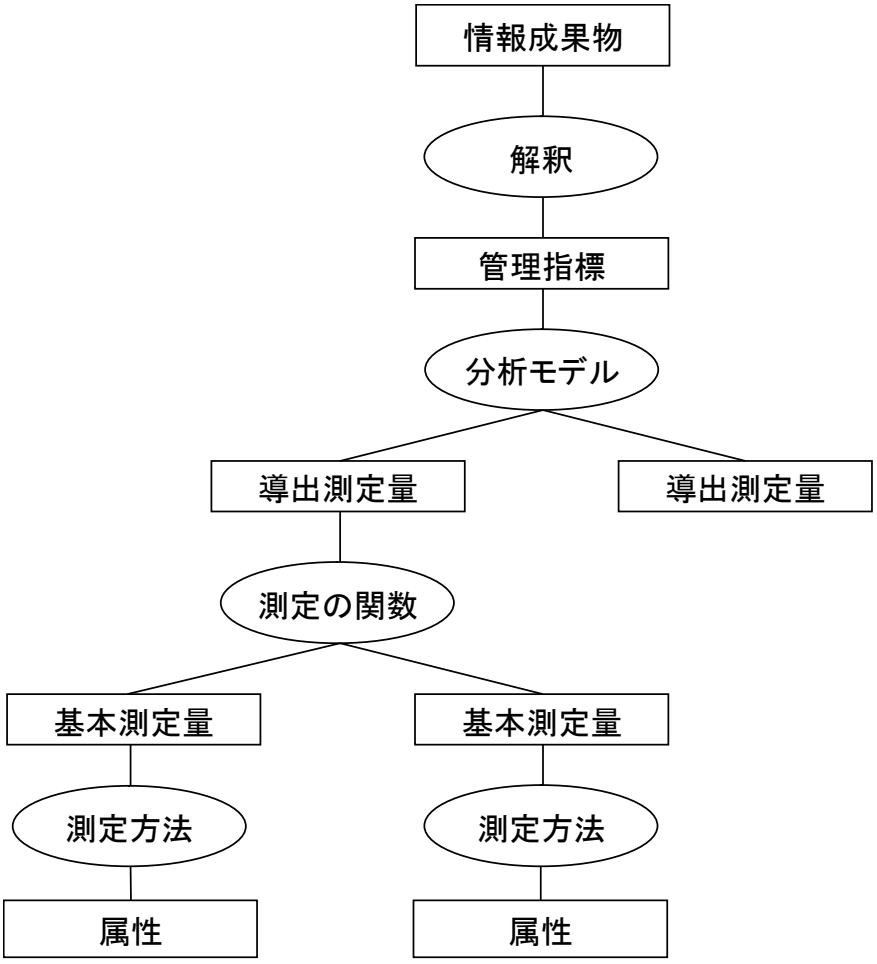

図 2.3　測定情報モデル（JIS X0141:2004 の図 A.1 をもとに作成）

解釈モデルに基づいて，管理指標が示す情報と判断基準とを照らし合わせることで，最終的な情報成果物を得ることができる．このようにして定量的測定プロセスの実施結果として得られる情報成果物に基づく意思決定は，プロジェクトを客観的に管理することを可能とする．

以降では，この測定情報モデルに基づいた概念構造，および用語を用いて議論を進める．

CMMI

CMMI はソフトウェアプロセスを管理・改善するための参照モデルである．CMU/SEI が 1991 年に発表した SW-CMM(Capability Maturity Model for Software) をもとに，その発展過程である SECM(Systems Engineering Capability Model)，IPD-CMM(Integrated Product Development Capability Maturity Model) など複数のプロセス成熟度モデルを統合し，開発された．CMMI では，CMMI 登場以前に問題となっていた複数のプロセス成熟度モデルを用いる場合に生じる不一致や重複などが解消されている．また，ISO/IEC15504 に準拠して開発されており，組織の持つプロセス能力を評価し，改善の指針を与えるためのモデルである．

CMMI では，組織の成熟度を表す形式として，段階表現と連続表現の 2 つが用意されている．どちらの表現方法においても，プロセス領域と呼ばれる，ある領域において関連する改善活動の集合を要素としてモデルが構成されている．CMMI を用いたプロセス改善では，組織がプロセス改善活動の目的に従い，段階表現または連続表現のどちらかを選択する．

段階表現では，図 2.4 のように，組織全体のプロセス成熟度を 1〜5 の 5 段階で表現する．各成熟度では，プロセス領域ごとに達成すべき事項（ゴール）が定められており，それぞれの段階で定められたゴールを全て達成することでその段階のレベルに至ったと判定される．これにより，次の段階へ向けて達成すべき課題が確立されるため，堅実な改善が見込まれる．

連続表現では，図 2.5 のように，プロセス領域ごとに 0〜5 の 6 段階の能力レベルを表現する．このように表現することで，プロセスの中で特に問題がある点を特定でき，問題改善を集中的に行うなどの弾力的な改善策を講じるための目安として有

用である．

CMMI では，段階表現の成熟度レベル 3「定義された」段階に到達するための条件として，プロセス領域「組織プロセス定義（OPD：Organizational Process Definition）」における改善活動に取り組むことが求められている．このプロセス領域の目的は，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であるとされている．そして，そのためには**組織標準のプロセス**と**テーラリング指針**が必要である．

ここで，組織標準のプロセスとは，どのようなプロジェクトでも適用できる形で，組織で定められた，抽象度の高いソフトウェア開発プロセスのことを指す．組織標準プロセスでは，具体的な作業内容や作業間の関連などが定められている．また，CMMI でのテーラリングとは，プロジェクトの特性に合わせて，組織標準プロセスからプロジェクトで必要なプロセスを取捨選択・調整し，プロジェクトごとの開発計

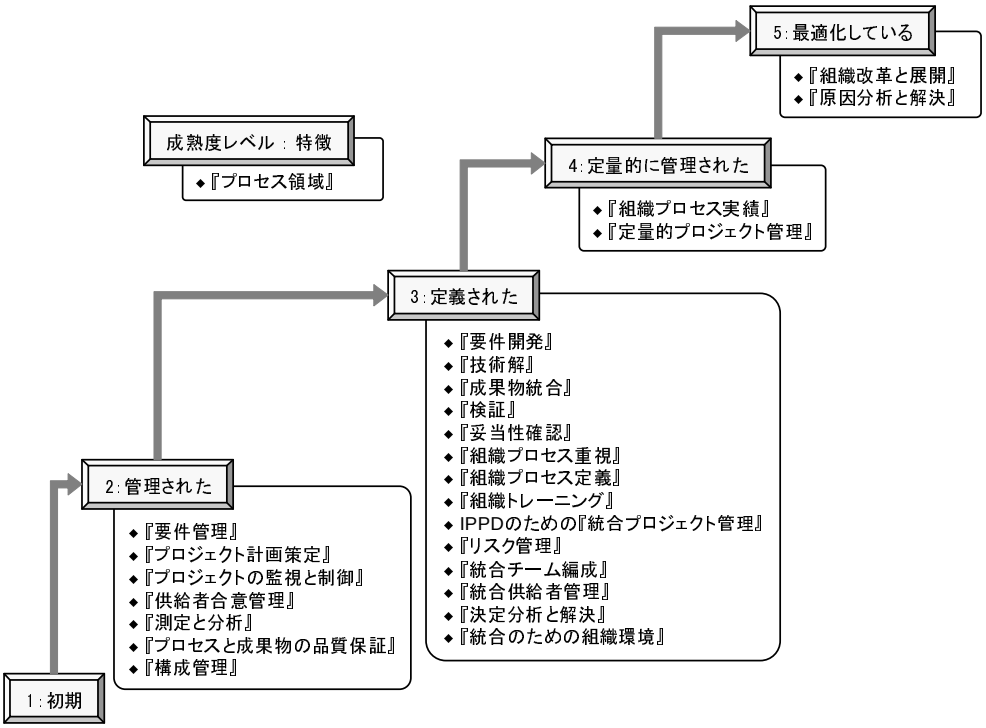

図 2.4　CMMI の段階表現の構造

画を作成する作業のことを指す．テーラリング指針とは，テーラリングの具体的な作業手順を定義したものである．

このようなテーラリング指針の具体的な内容や手順は，各々の組織で別途定義するものであり，多くの部分をプロジェクト計画者の経験に依存しているのが実状である．

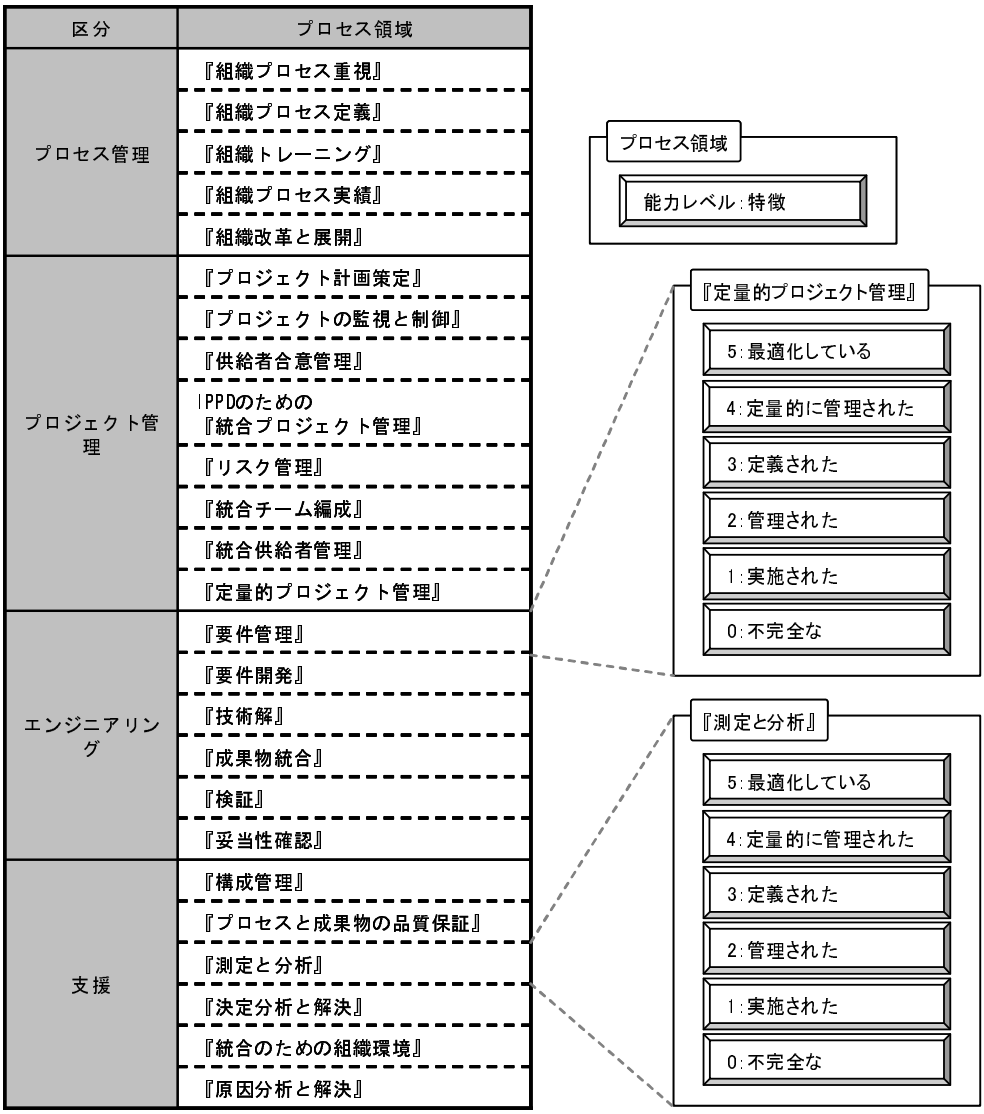

図 2.5　CMMI の連続表現の構造

2.2.2. ソフトウェアプロジェクトの計画立案に関する研究

開発プロジェクトの初期に行う計画立案作業では，ソフトウェアの開発プロセスと，その管理プロセスの構築を行う．計画立案作業は，主に組織レベルもしくはプロジェクトレベルでのプロセス構築を行うオーサリング作業と，あらかじめ定義されたプロセスを個々のプロジェクトの状況に沿って修整するテーラリング作業に分割できる．ここでは，オーサリング作業およびテーラリング作業に関する研究について述べる．また，これらの作業を実施する際に参照する，電子プロセスガイドに関する研究についても述べる．

プロセスのオーサリングに関する研究

Becker-Kornstaedt らは，プロセスモデルのオーサリングを支援するための環境である Spearmint を開発している [10]．プロセスモデルとは，開発プロジェクトにおける要件定義から開発完了までの活動を，時間軸に沿って定義したものである．Spearmint は，プロセスモデルを様々な視点からの「ビュー」を用いて表示したり編集したりすることで，ソフトウェアプロセスの理解と意思疎通を改善することを目的としている．ビューは，成果物と作業・役割の関連を表したプロダクトフロービュー，成果物と作業の持つ階層構造をツリー形式で表示した分解ビュー，各要素の持つ属性が一覧表示する属性ビュー，プロセスを詳細なテキストで表したテキストビューの 4 つが存在する．また，Spearmint は簡易 Web サーバ機能を持つため，Web ブラウザを用いて，モデリングしたプロセスに関する知識を組織内で共有することができる．

Eclipse Foundation は，プロセスのオーサリングとテーラリングを支援する環境である EPFC（Eclipse Process Framework Composer）を開発している [73]．EPFC の目的は，プロセスエンジニアによるプロセスの選択とオーサリング・テーラリングを支援すること，プロセスの実行者にプロセスに関する知識を与えることの 2 つである．EPFC はその特徴として，プロセスの目標や作成する成果物，ロールなど，プロセスのメソッドに関する記述を個々の作業内容や開発ライフサイクルなど，プロセスのコンテンツを表す情報と分離して扱うことができる．これにより，新たなプロセスを構築する際は，既存のプロセスを再利用することができ，既存のプロセス

をプロジェクトに合わせてテーラリングする際も，基底となる要素に影響を与えることなく特定のプロセスを修正することが可能となる．

プロセスのテーラリング支援に関する研究

テーラリング支援に関する研究として，Basili らは，プロジェクトの目的・制約に合わせてプロセスをテーラリングする手法と，プロセスの改善を実現する手法を提案し，それらに基づいたテーラリングツール環境 TAME を構築している [8]．TAME は計測ツールを統合した環境であり，開発プロセスのテーラリングおよび改善活動を支援する．適用組織は TAME を導入することにより，効率の良いテーラリングの実施が期待できる一方，テーラリング作業の自動化に必要なプロセスモデルを新しく導入することが求められる．

Park らは，テーラリングの支援を目的として，ニューラルネットワークを用いて，標準プロセスの中から各プロジェクトにて採用すべきプロセスをフィルタリングする手法を提案している [67]．Park らの手法を用いることで，テーラリング時にプロジェクトで必要なプロセスを，組織標準プロセスの中から絞り込むことができ，テーラリング作業が容易となる．

Huo らは，実際のプロジェクトのデータからプロセスパターンを抽出する手法を提案している [31]．Huo らの手法を用いて抽出したパターンを用いることで，テーラリング時に実際にプロジェクトで採用するプロセスの選定がより容易に行える．

これらの研究では，テーラリング時に有用な参考情報をプロジェクト計画者に提示することで，プロセステーラリングの支援を行っている．しかし，テーラリングの手順に関して，具体的な指針などは与えられておらず，プロジェクト計画者は自らの経験に基づき，プロジェクトの特性に合わせてテーラリングを行う必要がある．

電子プロセスガイドに関する研究

プロセスガイドとは，ある特定のプロセスに関して，そのプロセスを実行するのに必要な参照情報を記述したものである [47]．プロセスガイドは，紙ベースのものでも利用可能だが，電子媒体で提供する方が，検索などの利便性の点で優れている．このように電子媒体で提供されるプロセスガイドを特に電子プロセスガイド（Electronic Process Guide, EPG）と呼ぶ．電子プロセスガイドに関しては，ハイパーリンクな

どを利用できるウェブを利用したものが多く提案されている．以下では，ウェブを利用した電子プロセスガイドに関する研究を取り上げる．

Boehm らは MBASE[11] という包括的なプロセスに関する解説を行う電子プロセスガイド [76] を公開している．Moe らは中規模なソフトウェア開発組織において，開発者がどのように開発プロセスに関する電子プロセスガイドを利用しているか，また導入にあたってはどのような組織的要因が関わってくるかについて分析を行っている [55]．Jeffery らのグループは，EPG/ER[50] というプロセスに関する指針と知見をまとめた電子プロセスガイドを提案している．EPG/ER は，中小企業のプロセス改善やプロセス革新での導入実績が報告されている．

これらの研究では，テーラリング済みのプロセスを運用する上で有益な情報を提供しているが，テーラリングに対する具体的な指針や支援に関しては対象外となっている．計画の立案支援のためには，電子プロセスガイドによってテーラリングの参考となる情報を提示するとともに，テーラリング作業自体の支援を行うことが有益である．

2.2.3. ソフトウェア計測，品質評価に関する研究

定量データの収集支援に関する研究

ソフトウェア開発プロジェクトの作業記録を自動収集する研究は，近年盛んに行われている．これらのシステムの多くは，ソフトウェア開発で利用される，版管理システムやバグ管理システムなどと連携して動作する．ここでは，このような作業記録やそれを利用した定量データの収集を支援する研究について述べる．

EASE プロジェクトで開発された EPM（Empirical Project Monitor）[65] は，広く普及している開発支援ツール（版管理システム CVS[*1]，障害管理システム GNATS[*2]など）と連携し，開発履歴データをリアルタイムに収集し，定量データ分析を可能にするシステムである．EPM では，個々の開発支援ツールが収集したデータを集計，分析することで，グラフなどの形で可視化する．これにより，利用者は開発プロジェクト実施時に，開発者に作業負荷をかけることなく，プロジェクトの状況

[*1] http://www.nongnu.org/cvs/
[*2] http://www.gnu.org/software/gnats/

をリアルタイムに分析，可視化することが可能となる．

Johnson らのグループは，Hackystat を開発，公開している [53]．Hackystat は，開発者の行動に関するデータを収集することを目的としており，開発者の利用するツールに「センサ」と呼ばれるデータ収集のためのプラグインを導入することで計測を行う．Hackystat では，開発者が利用しているツール上での，「ファイルを開く」「カーソルを動かす」といった細かい粒度での作業を記録，収集することができる．

フラウンホーファー協会実践的ソフトウェア工学研究所など 7 組織が参加する SoftPit プロジェクト[*3]でも，定量的管理を支援する環境の提供を行っている．SoftPit では，Münch らが提唱する Software Project Control Center という概念 [59] に基づき，開発に関わる複数のステークホルダーが持つ，プロジェクト管理に関する種々の目的を扱うことを想定している．また，SoftPit では Specula というツールを開発している[*4]．Specula は，プロジェクトで収集されたデータを一箇所に集約し，多種多様な分析のためのビューを提供している．また，このビューはプロジェクトの管理目的に沿ってカスタマイズが可能となっている．

定量データの利用状況に関する研究

プロジェクト管理に用いる定量データに関しては，これまでに多くのものが提案されている．Riguzzi は，定量データの収集方法および分析手法について，設計工程，実装工程など，工程ごとに利用できる代表的な定量データについて具体例を交えて解説している [70]．Pfleeger らは，定量データの測定・分析に際して，研究者や実務家がどのようにして協調して作業を行えば良いかを示している [68]．

定量データの採用状況に関する実態調査としては，日本国内において，情報処理推進機構・ソフトウェア・エンジニアリング・センターによる収集状況調査 [82] や日本情報システム・ユーザー協会による調査 [83]，日本科学技術連盟による有効性評価に関する調査 [63] などが行われている．定量的管理を含む，プロジェクトにおけるプラクティスの導入状況の調査としては，Cusumano らによる調査が挙げられる [16]．ソフトウェア開発組織への定量的管理の導入に関しては，Gopal らの研究が挙げられる [23]．この研究では，ソフトウェア開発プロジェクトにソフトウェア測定活動

[*3] http://www.soft-pit.de/

[*4] http://code.google.com/p/specula/

を取り入れる際に，組織内部および外部の環境が導入に与える影響に関する調査を行っている．また，ソフトウェア開発組織内における定量的管理の実践については，岩切らの報告が挙げられる [45]．岩切らは，まず定量的管理を実践するために関連する組織内プロセスの改善を行った．その上で定量的管理を実践し，その過程で発生した改善を必要とする事例をまとめ，どのように改善を行ったかを報告している．また，定量的管理を実践した結果，上流工程での品質確保，出荷後の品質向上が達成されたと報告されている．

ソフトウェアの評価手法に関する研究

ソフトウェアの品質評価は，大別してプロダクト評価とプロセス評価の 2 つがある．プロダクト評価は，ソフトウェア製品そのものやソフトウェアの振る舞い等の属性を測定し，評価する手法である．一方，プロセス評価は，ソフトウェアの開発過程，すなわちソフトウェア開発プロセスの品質に関わる属性を測定，評価する手法を指す．

プロダクト評価に関する手法としては，ソースコード行数やファンクションポイントといったソフトウェアの規模による評価 [29]，オブジェクト指向言語でのクラス間の関連に基づく評価 [7]，ソースコードの複雑度による評価手法 [71]，ソースコード中に含まれる類似コード片（コードクローン）とソフトウェアの信頼性，保守性との関連に関する研究 [56] などが挙げられる．また，ソフトウェア仕様書や設計文書のレビューに関する研究 [51] や，ソフトウェア中に含まれる欠陥数の予測の研究 [75] も，このようなプロダクト評価手法に関する研究と捉えることができる．

一方，開発プロセスの分析・評価は，ソフトウェア開発において重要となっており，これまでに多くのパラダイムが提案されている．例えば，Basili らの提唱する GQM (Goal - Question - Metrics)[9] や McGarry らの提案する PSM (Practical Software Measurement)[52] 等の方法論，カーネギーメロン大学ソフトウェア工学研究所が開発した CMMI (Capability Maturity Model Integration)[13] や Humphry が提唱した PSP (Personal Software Process)[30] などのフレームワークが存在する．このようなプロセスを評価する取り組みの多くは，組織のプロセス実施能力を定性的に評価するものか，成果物の管理情報として記録されたデータからプロセスに関する情報を取り出し，作業効率や生産性についての定量的な評価を行うものである．

2.3. 定量的管理を実施する際の課題

2.2 節で述べたように，ソフトウェア開発における定量的管理に関して，これまでに多くの規格やフレームワーク，方法論が提案されてきた．特に，データ収集に関しては，版管理システムに代表されるような開発支援ツールを利用した収集が多くの組織で行われていると考えられる．その一方で，定量的管理を実践するにあたっての計画策定や，収集したデータの分析，評価，さらにはその結果を利用したプロセスの改善活動については，十分な支援がなされていない．

先に示した定量的管理を実施する 4 つのアクターごとに，実践にあたっての課題を以下に示す．

プロセスエンジニア

標準開発プロセス定義と管理指標をオーサリングする際には，構築するプロセスや定量データ，プロセスに関連する成果物が，組織の実状を適切に反映する必要がある．しかし，数十から数百の要素で構成された標準開発プロセス定義や管理指標に対し，これらを考慮してオーサリングを行うのは困難である．

プロジェクト計画者

開発管理計画のテーラリングを行う際には，定量データの測定・分析作業が開発プロセスの各部に分散して組み込まれるため，開発プロセスおよび管理指標の全体構造を深く理解する必要がある．また，プロセスエンジニアと協調してプロジェクトに合わせて標準開発プロセス定義や管理指標を修正する際は，修正された開発プロセス定義・管理指標に矛盾や無理が無いことを確認する必要がある．そのため，開発プロセスと管理指標のテーラリングを実施するためには，多くの経験を要する．

開発者・測定者・分析者

データの収集・分析を行う際，管理指標とそれに関連する定量データに関して，その収集・分析方法に関して深い理解が必要である．定量データや管理指標についての理解が不十分であると，計画時に定められたデータが正しく収集されないおそれや，

管理そのものが形骸化する危険性がある．また，どのような手法を用いて，分析，評価を行い改善につなげていけば良いかという手法も明らかになっていない．

上記のように，定量的管理を実践するには，いくつかの課題があると考えられる．これらの課題に対する解決策を検討するにあたっては，実際にソフトウェア開発組織において，どのようなプロセス管理が行われているかを確認し，どのような課題があるかを確認する必要がある．そこで，次章では，定量的管理の実態を明らかにするために行った調査について報告する．

第 3 章

管理指標を利用した定量的管理の実態調査

本章では，日本国内の大規模ソフトウェア開発組織（株式会社日立製作所のエンタープライズソフトウェア開発部門．以下，調査対象組織と略す）で実施した，管理指標を利用した定量的管理の実態調査について報告する．第 1 章で示した通り，定量的管理が実際にどのように実施されているのかは十分に明らかになっておらず，実施にあたっての課題も明らかになっていない．本章で報告する調査では，特に管理指標を用いた定量的管理を行っているソフトウェア開発組織を対象とした．この調査は，定量的管理を実践する上で，実際の現場ではどのような管理が行われているのか，また管理を実施している際にどのような問題が生じているのか，問題が生じている場合にはどのように問題を回避して管理を行っているかを明らかにするために行った．

3.1. はじめに

定量的管理を実施することで，開発プロセスの実行中に問題を特定し改善することができる．しかし，実際のプロジェクトにおいて，定量データの収集や調整を行うためには多大なコストがかかる．このため，プロジェクトを成功に導くためには，有効に活用されていないデータに関しては，管理指標定義の見直しを行うことが望ましい．そのためには，定量的管理を実践している企業では管理指標の運用状況や管理指標利用に必要な定量データの収集状況を明らかにする必要がある．これまでに，

定量データの収集方法や分析手法に関する調査 [68, 70] や，開発組織での採用状況に関する調査 [82, 83] は行われている．しかし実際の企業でのソフトウェア開発において管理指標を用いた定量的管理がどのように実施されているかを調査した研究は，我々の知りうる限り存在せず，その実態は明らかになっていない．運用実態を明らかにした上で，運用が困難な管理指標が存在する場合には，プロジェクトもしくは組織のどのような要因が管理指標の運用を妨げているかを明らかにし，しかるべき対策を講じる必要がある．

本章では，調査対象組織のあるソフトウェア開発担当部署において，管理指標がどのように運用されているかを調査した．調査対象組織を選定した理由は，調査対象部署において組織標準の開発プロセスを定義しており，定量的管理についても組織的に導入を推進しているためである．そして，プロジェクト管理者（プロジェクトマネージャ）を対象に社内で定義されている管理指標を個々のプロジェクトでどのように運用しているか，ヒアリングを行った．また，個々の指標がプロジェクトを管理する上で重要なものであるか，プロジェクトマネージャの主観的評価も収集した．

その結果，多くの管理指標が大多数のプロジェクトの管理に利用されている一方，組織内で基準値が整備されていないものやデータの収集体制が整っていないものなど，一部の指標は適切に運用されていないことが判明した．また，指標を運用する際には，組織で定義された通りに指標を運用するだけでなく，プロジェクトの状況にあわせて調整を行っていることが明らかになった．本章では，この調査結果を報告するとともに，それをもとにソフトウェア開発組織が定量データに基づく管理指標を用いた定量的管理手法を導入する際の指針を示す．

以下，3.2 節でソフトウェア開発プロセスの定量的管理と課題について概説し，3.3 節および 3.4 節で管理指標・定量データに関する利用実態調査について報告する．3.5 節では 3.4 節の結果から，今回調査対象とした開発組織における利用実態を明らかにし，ソフトウェア開発組織において定量的管理を実践する際の留意点についてまとめる．最後に 3.7 節でまとめを述べる．

3.2. 定量的管理実施に関する実態調査を行う上での着目点

3.2.1. 定量的管理

定量的管理を実践する際，単に定量データを収集するだけではなく，収集した結果を常に評価してはじめて，プロジェクト管理をうまく進めることができる．そのためには，プロジェクトの状況をどのような狙いで測定するかという「目的」と，その目的が達成できているか確認するための「管理指標」を定める必要がある．管理指標は，定量データを一定の手法に従って分析することで求めることができる．プロジェクト管理者は，管理指標を利用することでプロジェクトを管理し，場合によっては是正のための措置をとるなどの 意思決定を行う．

例として，要件定義作業の管理を考える．管理者が「作業の進捗を把握し，計画との差異を確認する」という目的を立てたとする．そして，目的が達成できているかを確認するため「要件定義の完成したドキュメントページ数の推移」という管理指標を設定したとする．この管理指標を求めるためには，「要件定義の完成したドキュメントページ数」という定量データを定期的に測定する必要がある．そして，測定結果を時系列に分析することにより，ドキュメントページ数の推移を求めることができる．この管理指標を見て，計画より遅れている場合には，要件定義作業に人員を追加するなどの意思決定を行う．

情報処理推進機構 ソフトウェア・エンジニアリング・センター（IPA/SEC） では，組織間で標準的に利用される一般的な管理指標群を，測定項目リストとして公開している．このリストの一部を表 3.1 に示す．このリストでは，管理指標とその測定方法，分析方法などが整理されている．個々の指標については，プロジェクトの状況をどのような狙いで測定するかという「目的」，目的実現のために必要となる「ベース尺度（定量データ）」とその「測定方法」，および定量データを測定する「収集者」，測定された定量データを分析する「利用者」などが記述されている．

表 3.1 の管理指標群は，組織間で標準的に利用できるよう一般化されているが，実際にはそれぞれの組織や部署の事情にあった管理指標群を用意する必要がある．一般的に，個々の開発組織では，プロセス資産として複数のプロジェクトで利用できる

表 3.1 管理指標群の例

項目番号	知識エリア	測定の目的	利用者	ベース尺度	尺度の収集元・方法	収集者
T1	タイム	PM,PMO	要件定義完成したドキュメントページ数の推移	1. 要件定義の完成したドキュメントページ数	WBS，要件定義書で確認する．定期的（週次）に 1. を収集する．	要件定義者
R2	リスク	リスクの時系列変化を測定し，問題の早期解決を図る．	PM,PMO	1. 一定期間のリスクの発生率 2. 増減	発生状況レビュー時，収集	

形で，「組織標準の管理指標群」をあらかじめ定めている．これにより，管理目的の設定や定量データの選定など，定型的で非常に手間のかかる作業を簡略化し，効率よくプロジェクト管理を行うことができる．

3.2.2. 実態調査を行う上での着目点

定量的管理を行う際に収集すべき定量データの項目はこれまでに数多く提案されており [68, 70]，多くの開発組織で実際に収集されている．例えば，IPA/SEC が 2008 年度に実施した調査 [82] では 20 社 2056 プロジェクト，日本情報システム・ユーザー協会（JUAS）の調査 [83] では 93 社 341 プロジェクトを対象に，定量データの収集状況に関する調査が行われている．文献 [83] の調査によれば，例えばコード行数については 341 プロジェクト中 183 プロジェクトが何らかの形で計測していることが明らかになっている．

一方で，日経コンピュータが実施した調査では，何らかの定量的管理手法を導入している組織であっても，そのプロジェクトの成功率は 45.6% であることが報告されている [60]．この原因としては，組織内において定量的管理の運用に多くの障害があるため定量的管理を適切に活用できていないことが考えられる．実際に，小笠原らによって実施された定量データの有効性に関する調査（58 プロジェクト，66 定量データが対象）[63] では，有効性が低いものや収集作業が実施困難なものが含まれていることが報告されている．定量的管理を活用してプロジェクトを成功に導くためには，まず定量的管理が開発組織においてどのような形で運用されているのかを十分に明らかにする必要がある．そして，その上で運用が困難な管理指標が存在している場合には，障害がどのような要因で発生しているかを明らかにし，対策を講じる必要がある．

そこで本研究では，管理指標を用いた定量的管理を実施しているソフトウェア開発組織を対象に，実プロジェクトにおいて定量データを用いた定量的管理がどのように実施されているかを調査した．

本研究では管理指標群の運用形態に着目し，以下の観点から調査を行った．

- 管理指標を利用した開発管理は，どのような業種向けの開発や，どの開発フェーズで導入されることが多いか．
- 組織標準の管理指標を利用した開発管理を導入する場合，開発部署や開発プロジェクトに合わせてどのような調整が必要か．
- どのような管理指標の導入が困難か．

これらの観点から実際の開発現場における管理指標の運用の状況を明らかにすることで，管理指標を開発プロジェクトに導入して定量的管理を行う場合に，どのようなことを考慮する必要があるのかを明らかにできると考えられる．

3.3. 管理指標・定量データの利用実態に関する調査

3.3.1. 実施概要

調査対象組織において策定されている管理指標群がどのように利用されているか，3.2.2 節で示した観点からヒアリングを行った．ヒアリングは，調査対象組織で遂行されている実プロジェクトの管理者を対象として実施した．その内容は，どのような管理指標を利用しているか，管理指標が利用されていないのはどのような理由か，の 2 点を調査するものである．以降，3.3.2 節で調査対象組織，3.3.3 節で調査対象とした管理指標の詳細を示す．また，3.3.4 節で，今回実施したヒアリング項目の決定指針について述べる．

3.3.2. 調査対象組織・プロジェクト

調査対象組織は，CMMI 段階表現における成熟度レベル 3 相当（組織標準プロセス定義を利用）で，かつ，定量的管理についても組織的に導入を推進している．

本調査では，13 件のプロジェクトを対象に調査を実施した．調査対象となったプ

ロジェクトは，大きく分けて「金融・保険・証券」「公共機関」「流通・製造業」の 3 種類の業種向けのシステム開発プロジェクトに分類される．これらのプロジェクトの中には，現在進行中のプロジェクトも含まれる．表 3.2 に回答のあったプロジェクトの対象業種と状況，規模を示す．

表 3.2　回答者と回答者の担当するプロジェクトのプロフィール

	対象業種	プロジェクトの状況	プロジェクト規模（k ステップ）
(a)	金融・保険・証券	進行中	600
(b)	金融・保険・証券	進行中	700
(c)	金融・保険・証券	進行中	900
(d)	金融・保険・証券	進行中	600
(e)	金融・保険・証券	完了	400
(f)	公共機関	完了	6,000
(g)	公共機関	完了	2,500
(h)	公共機関	進行中	1,000
(i)	公共機関	進行中	300
(j)	流通・製造業	進行中	4,500
(k)	流通・製造業	進行中	700
(l)	流通・製造業	進行中	200
(m)	流通・製造業	進行中	2,000

3.3.3.　調査対象の管理指標

調査対象組織ではプロセス改善を目的として表 3.1 と類似した形式で，プロジェクトの管理指標群が定義されている．その管理指標群は，従来社内で実践されてきた開発管理手法をもとに，CMMI ベースによる組織改善を進める上で新たに提案したものを加えて定義された．指標の中には，その有効性や定義の適切さが十分確認されていないため，現在導入を検討中のものや，一部のプロジェクトでのみ導入されているものも含まれている．

また，指標の中には指標を利用して意思決定を行う際の基準値や目標値が整備されていることが前提とされているものがある．従来調査対象組織内で実践されていた管理手法と同様の指標に関しては，これらの値は整備されている．一方で，新規で

追加された指標の中には，このような基準値や目標値が整備されていないものも含まれる．

今回はこれらの管理指標群のうち，個々のプロジェクトでの利用を想定している 47 指標を調査対象とした．調査対象の管理指標群は，7 つの観点からその利用目的が整理されている．表 3.3 に指標番号と利用目的による分類の対応を示す．なお，個々の指標の概要については，付録の表 B.1 から表 B.5 に示す．

表 3.3　指標番号と指標利用分類の対応表

指標番号	利用目的	管理対象
#1〜#21	進捗管理	プロジェクトの進捗
#22〜#27	レビュー	レビュー作業効果
#28〜#32	テスト	テスト結果の妥当性
#33〜#36	プロセス品質保証	開発プロセスの実施状況
#37〜#39	リスク管理	プロジェクトで発生したリスク
#40〜#43	要件管理	顧客からの要件
#44〜#47	支援プロセス	調達管理・構成管理

3.3.4. ヒアリングシートの設計

ヒアリングでは，プロジェクトの業種による指標利用傾向の差異を確認するため，プロジェクトが対象とする業種，規模および管理指標の評価を収集した．また，具体的にどのような利用目的の指標が利用されているのかを確認するため，47 個の指標ごとに「指標利用の状況」と「重要度」の観点から，プロジェクト管理者に評価を依頼した．

指標利用の状況　指標をどのように利用したかを回答する．回答は「指示通りに利用した」「調整して利用した」「利用しなかった」の 3 つから選択する．また，指標を調整した場合は調整内容を，利用しなかった場合にはその理由をそれぞれ記入する．

重要度　指標に設定された目的がどの程度重要であるかを，5 段階で評価する．

指標利用の状況を分析することで，各プロジェクトでどの指標をどのように利用したか確認する．また，重要度を分析することで，指標定義が正しく理解されているかどうかを明らかにする．

また，上記の 2 項目に加えて，どのような意図で回答者が指標を評価したかを分析するために，指標に対する自由コメントも記入させた．このように，今回実施したヒアリングでは，利用の可否を議論するだけでなく，具体的な利用方法や利用しなかった場合の判断理由を詳細に分析するため，回答者のコメントをできるだけ多く取得するように設計した．また，ヒアリングした結果得られた回答（コメント）を分析し，その回答内容の中で，不明な点については，追加ヒアリングを実施し，回答内容の真意について改めて聞き取りを行った．

3.4. 調査結果

3.4.1. 指標の傾向

図 3.1 の横軸は対象プロジェクトにおいて各指標を採用したプロジェクトの個数，縦軸は指標番号を示している．図 3.1 では，個々の指標ごとに，「指標の定義通りに利用した（指示通り）」プロジェクトの数と，「指標定義をプロジェクトの特性に合わせて一部修正して利用した（調整）」プロジェクトの数を，プロジェクト対象業種ごとに分類している．また，表 3.4 に管理指標群ごとの指標の平均利用率および平均

表 3.4 指標群ごとの利用率・調整率

指標群	利用率				調整率			
	金融	公共機関	流通・製造業	平均	金融	公共機関	流通・製造業	平均
進捗管理	0.75	0.89	0.86	0.83	0.10	0.35	0.13	0.20
レビュー	0.33	0.71	0.42	0.47	0.00	0.49	0.00	0.24
テスト	0.96	0.90	1.00	0.95	0.00	0.28	0.05	0.10
プロセス QA	0.55	0.75	0.44	0.58	0.29	0.33	0.08	0.27
リスク管理	0.47	0.67	0.83	0.64	0.00	0.50	0.17	0.27
要件管理	0.55	0.88	0.75	0.71	0.21	0.38	0.44	0.33
支援プロセス	0.20	0.50	0.75	0.46	0.50	0.50	0.54	0.56
平均	0.62	0.81	0.81	0.72	0.13	0.38	0.16	0.24

調整率を示す．ここで，指標の利用率は，調査対象のプロジェクトのうち，指標を利用した（定義通りに利用した，もしくは調整して利用した）プロジェクトの割合である．また，指標の調整率は，指標を利用したプロジェクトのうち，指標を調整して利

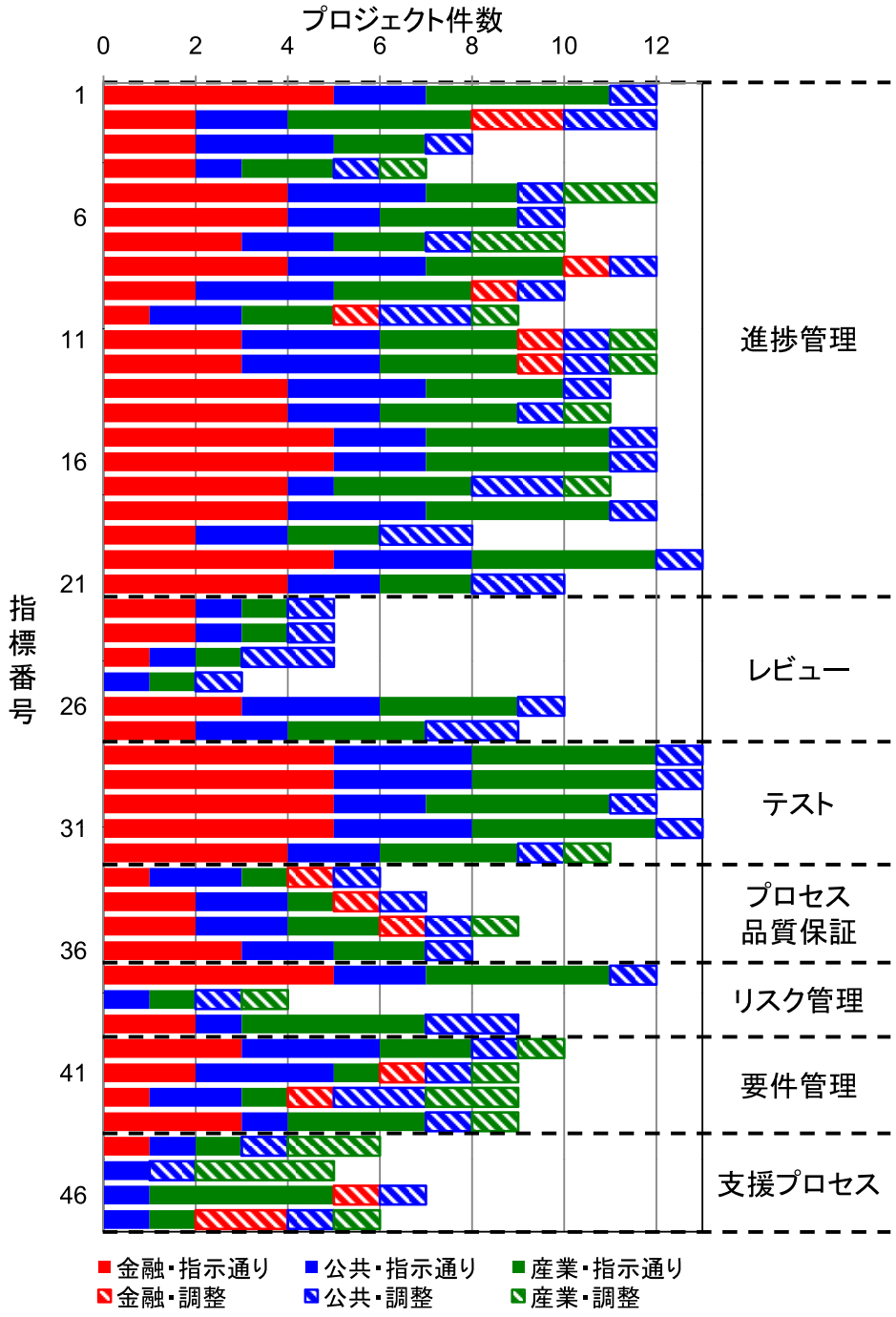

図 3.1　指標別の採用状況

用したプロジェクトの割合である．平均利用率，および平均調整率は，各指標の利用率・調整率を算出し，利用目的による分類および業種による分類ごとにその平均値を求めた．

図 3.1 および表 3.4 より，進捗管理，テスト，要件管理に関する指標は，半数以上のプロジェクトにおいて，利用されていることがわかる．一方で，レビュー，プロセス品質保証，リスク管理，支援プロセスに関する指標の一部は，半数以上のプロジェクトで採用が見送られている．

図 3.2 は重要度の 5 段階評価について，回答をまとめたものである．縦軸は指標番号，横軸はそれぞれの指標に対する評価を段階ごとに積み上げたものである．

図 3.2 より，進捗管理，テスト，要件管理の指標群に属する指標の目的について 4 以上の評価をした管理者は半数以上存在した．一方で，レビュー，プロセス品質管理，支援プロセスに属する指標の目的について，一部を除き，3 以下の評価をした管理者が半数以上存在した．

表 3.5 コメントの分類

	進捗管理	レビュー	テスト	プロセス品質保証	リスク管理	要件管理	支援プロセス	合計
指標の重要性へのコメント	2	0	0	0	0	0	0	2
目的・定義の問題点の指摘	52	20	5	13	6	6	14	116
調整した理由の説明	41	5	5	4	1	12	4	72
利用しなかった理由	39	37	1	21	8	15	20	141
その他	9	2	3	6	1	1	5	27
合計	143	64	14	44	16	34	43	358

表 3.6 調整して利用した指標に関するコメントの分類

	進捗管理	レビュー	テスト	プロセス品質保証	リスク管理	要件管理	支援プロセス	合計
定量データの一部を取捨選択する	16	2	0	0	0	0	1	19
代替となる定量データを利用する	5	2	1	0	1	7	3	19
測定頻度・単位を変更する	9	1	3	3	0	2	0	18
他社作業に合わせて調整	10	0	0	0	0	0	0	10
不明	2	0	1	1	0	5	0	9
合計	42	5	5	4	1	14	4	75

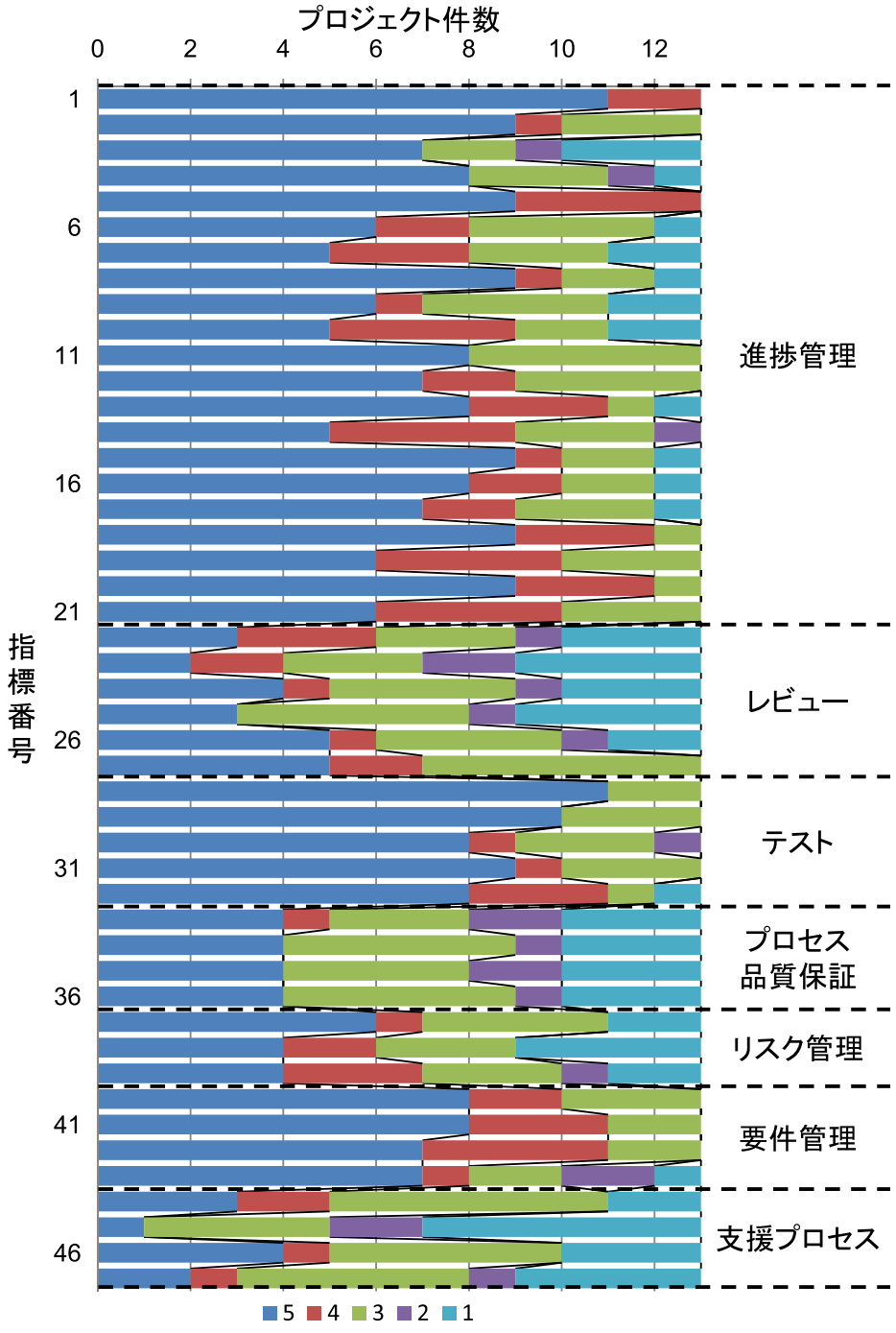

図 3.2　指標別の目的の重要度

3.4.2. 管理指標に対するコメント

ヒアリングで得たコメントを分類した結果を表 3.5 に示す．コメントは，指標そのものに対するコメント（指標の重要性へのコメントや，問題点の指摘）と指標の利用方法に関するもの（調整した理由の説明や利用しなかった理由）に大きく分類できる．指標そのものへのコメントとしては，指標を運用する上での問題点や，目的や定量データなど指標の定義に関する問題点の指摘，プロジェクト管理において指標を利用して管理することの重要性に関するコメントが記述されていた．また，指標の利用方法に関するものとしては，調整して利用した際の具体的な調整内容や，利用しなかった際の判断理由が記述されていた．

今回の調査では，この中でも特に指標の運用形態に着目して議論するため，「指標を調整して利用した際の具体的な調整内容」と「指標を利用しなかった際の判断理由」に関するコメントに着目する．表 3.6 に指標の調整内容に関するコメントの分類，表 3.7 に利用しなかった際の判断基準に関するコメントの分類を示す．なお，表 3.6 に関しては，1 人の回答者が 1 つの指標に対して複数の調整方法を施している場合がある．

調整して利用した指標に関しては，「定量データの一部を取捨選択して利用する」(19 件)「定義されている定量データとは異なる，代替となる定量データを利用する」

表 3.7 利用しなかった指標に関するコメントの分類

	進捗管理	レビュー	テスト	プロセス品質保証	リスク管理	要件管理	支援プロセス	合計
目的に応じた管理は行っている	17	9	0	2	1	7	9	45
違う手法を用いた管理を実施	0	3	0	0	1	0	0	4
定義されているデータでは管理できない	6	13	0	2	3	4	0	28
データ収集に手間がかけられない	5	1	0	0	0	0	0	6
指標運用の対象外	0	4	1	10	1	1	1	18
指標を利用する必要がない	10	0	0	5	0	0	9	24
指標の存在知らなかった	0	2	0	0	0	0	0	2
今後導入予定	0	0	0	2	0	3	0	5
使ってない（具体的な理由不明）	1	5	0	0	2	0	1	9
合計	39	37	1	21	8	15	20	141

(19 件)「定量データを測定する頻度・単位を変更する」(18 件) といった調整を施していることがわかった．この中で,「定量データの一部を取捨選択して利用する」「定義されている定量データとは異なる，代替となる定量データを利用する」という調整方法が，他の調整方法と比べて多い．調整されている指標群としては，進捗管理に関する指標と要件管理に関する指標が，他の指標と比べて調整されている割合が高いことがわかる．またその他に，進捗管理に関する指標を，共同で開発を行っている関連会社の作業状況に応じて随時調整しているプロジェクトが 1 件確認された．

利用されなかった指標に関しては,「指標自体は利用していないものの異なる手法で管理している」，という回答の割合が 45 件と一番多い．他には,「該当指標が対象としているプロセスがプロジェクトに含まれていないため，指標運用の対象外」(18 件) や「定義されているデータでは管理できない」(28 件) といった，指標の定義とプロジェクトの実態に乖離があると考えられる回答も多い．また,「管理指標を利用する必要がない」というコメントも 24 件あった．

3.5. 調査対象組織における利用実態の考察

本節では，前節の利用実態調査の結果をうけて，調査対象組織における指標利用の傾向について考察する．

3.5.1. 指標が利用される業種や開発フェーズ

図 3.1 より，進捗管理，テスト，要件管理に関する指標は，全体的に半数以上のプロジェクトで採用されていることがわかる．一方で，レビュー，プロセス品質保証，支援プロセスに関する指標の多くは，採用しているプロジェクト数が半数未満となっている．

これらの採用プロジェクト数が半数を切っている指標に着目して，指標に設定されている管理目的の重要度 (図 3.2) を見ると，どれも重要度の中央値は 3 以下と低い値を示している．このことから，採用プロジェクト数が少ない管理指標は，プロジェクトを管理するにあたってあまり重要ではないと考えられている可能性がある．なお業種に関しては，図 3.1 より，業種間で利用方法に大きな差異は確認されなかっ

た[*1].

3.5.2. 管理指標を用いた開発管理を導入するために要する調整作業

表 3.4 より，進捗管理，レビュー，プロセス品質保証，リスク管理に関する指標は，調整率が全体の平均と同程度であることがわかる．要件管理，支援プロセスに関する指標は，その他の指標と比較して，より調整される傾向にある．また，テストに関する指標は，平均よりも調整率が低い．図 3.1 より，指標#42（プロジェクトで対応が完了した要件数）のように，指標を定義通りに利用したプロジェクト数よりも調整して利用したプロジェクト数が多い指標も散見される．なお，これらの指標において，特定の業種だけが調整して利用している，といったような顕著な差は確認されなかった．

このように，支援プロセスに関する指標は，利用率が低いにも関わらず調整率が高い．一方で，進捗管理とテストに関する指標は，利用率が高い一方で調整率は低い．前者については，利用するために調整が必要になっているため，3.5.1 節にあるように，あまり利用されていないと推測できる．後者については，利用に関して調整が必要なく，いろいろな業種向けにそのまま利用できることが推測される．従って，組織標準の指標を作りやすく，また逆に各組織も比較的導入しやすいと言える．これ以外の指標については，ある程度の調整作業が必要であると推察される．

そこで，表 3.5 に挙げたコメントを詳細に分析し，どのような調整作業が行われたのかを検証したところ，表 3.6 のように，指標の調整には「定量データの一部だけを収集する」「定義されている定量データとは異なるものを利用する」「定量データを収集する頻度・タイミングを変更する」の 3 種類の方法が確認された．以下に，それぞれの調整方法について詳細を示す．

[*1] 業種によって利用方法（「定義通りに利用」，「調整して利用」，「未利用」）の傾向に違いがあるかを確かめるために，全指標に対して有意水準 5% でフィッシャーの正確確率検定を行った．その結果，有意確率 p が 0.05 未満の値となり，指標の利用方法と業種の間に関連が見られたのは指標#46（$p = 0.036$），指標#47（$p = 0.028$）の 2 つの指標だけであった．ただし，第 1 種の過誤を考慮すると，この 2 つの指標で誤って統計的に有意であるという結果が出たという可能性も考えられる．また，サンプル数の問題からも，この結果を議論することは難しい．よって，あくまでこの検定結果は，参考情報として付記するにとどめる．

定量データの一部だけを収集する

プロジェクトの中には，管理指標定義の中で定義されている定量データのうち，その一部のみを収集しているものがある．指標#11（サブシステムごとの開発規模の推移）では，「ドキュメントページ数」「データベース・ファイル規模（項目数）」「システムの画面・帳票本数」「プログラムコード行数」の 4 種類の定量データを収集するように定義されている．しかし，あるプロジェクトでは，このうち「システムの画面・帳票本数」「プログラムコード行数」の 2 種類の定量データしか収集していないと回答があった．

コメントでは，このような調整を行った理由についての明確な言及はなかった．しかし，指標そのものを利用しなかった理由としては，「収集に手間をかける価値がない」「指標の重要性は理解するが，人手が足りず割愛」といったコメントが得られた．このことから，このような定量データの取捨選択は，プロジェクトマネージャがプロジェクトの状況（規模や予算，納期など）を勘案した上で行っていると考えられる．例えば，ある管理指標が重要と考えるにも関わらず，収集にコストをかけられない場合が考えられる．また，複数の定量データを利用するような指標の場合，一部の定量データの定義が組織の作業実態に合っておらず，収集できていないケースも考えられる．

定義されている定量データとは異なるものを利用する

定義通りの定量データを収集せずに，個々のプロジェクトの判断で代替となるデータを収集しているケースもあった．例えば，指標#43（要件変更として認めた要件件数に対する仕様変更管理票，プログラム変更票の推移）では，定量データの一つとして，仕様変更管理票件数を収集することになっている．しかし，あるプロジェクトでは仕様変更管理票ではなく障害管理票の件数を利用していた．

このように，代替となる定量データを利用している理由は，先に挙げた定量データの一部のみを収集する理由と同じと考えられる．すなわち，管理指標の利用目的を重要と考え利用したいが，定義されたデータを収集するとコストがかかるため，よりコストのかからない別の定量データを利用していると考えられる．よって，代替となる定量データを利用しているプロジェクトが多い指標については，代わりに利用

しているデータについての妥当性について勘案した上で，組織標準の指標定義を修正していく必要がある．

定量データを収集する頻度・タイミングを変更する

定量データによっては，指標定義とは異なる頻度，タイミングで収集される場合があった．指標#35（作業改善指示書のカテゴリ毎の是正率と，要因別事故発生件数）は，プロセス品質保証検査の実施状況とその効果を測定するために利用される．指標定義では，指標の利用に必要な定量データをプロジェクト完了時に収集することになっている．しかし，プロジェクト完了時ではなく，プロジェクトの進行中に測定し状況を把握しているプロジェクトがあった．

このように，プロジェクトをより短い周期で管理するために，定義とは異なる頻度・タイミングで定量データの測定を行う場合があると考えられる．また逆に，定量データの収集コストを削減するために，設計工程時に毎日収集することになっているデータを週次で収集するようにするなど，測定の周期を長くとる場合もあると考えられる．

このように，定量データの内容によっては，測定するタイミングがプロジェクト毎に異なっていて，画一的に決められるものではないと考えられる．そのような場合には指標定義をあらかじめ調整できるように定めておく必要がある．

調整作業に関するまとめ

調査の結果，調査対象組織においては，指標の大部分が多くのプロジェクトで利用されていた．その一方で，プロジェクトによっては有効に活用されていない指標や，調整されて利用される指標が確認できた．調整されている指標に関しては，「定量データの一部だけを収集する」「定義されている定量データとは異なるものを利用する」「定量データを収集する頻度・タイミングを変更する」という 3 種類の調整方法をとっていることが明らかになった．

3.5.3. 利用を敬遠される指標

今回調査対象とした組織においては，基本的に全ての指標がプロジェクトの定量的管理を実践する上で必要なものとして定められている．しかし実際には，レビュー，プロセス品質保証，支援プロセスに関する指標のように，あまり利用されていない指

標が存在する．このような指標については，「指標の定義そのものに問題がある」のか「指標の利用者（プロジェクト管理者）に問題がある」のか，つまり，導入する側にとって，「指標を利用することの必要性が低い」のか「指標を使って管理を行うための障壁が高いと判断される」のかを明らかにする必要がある．そこで本節では，実際に利用率が低かったレビュー，プロセス品質保証，支援プロセスに関する指標および，リスク管理の#38 について，ヒアリングに記述されたコメントをもとに，その原因について考察する．

レビューに関する指標群

図 3.1 から，指標#26（種類別欠陥数の調査），#27（レビュー時の指摘項目に対する是正状況の把握）を除いて，この指標群に属する管理指標は利用されない傾向にあることがわかる．

表 3.5 から，レビューに関する指標群についてのコメント 64 件中，目的・定義の問題点の指摘が 20 件あったことがわかる．これらのコメントは，指標の目的や利用することによって得られる効果がわからないというものであった．また，利用しなかった理由を説明したものが 37 件に上っている．その内訳を表 3.7 のレビューの列に示している．利用しなかった理由として最も多かったのは，定義されているデータでは工程の管理が十分にできないというものであり，37 件中 13 件であった．また，複数の回答者が測定結果の妥当性を判断する具体的な基準がないとコメントしている．

これより，この指標群に属する指標が使われない理由は 2 つに分類することができる．一つ目は，多くの管理者が回答しているように，測定した結果の妥当性を判断する具体的な基準がないという理由である．例えば，レビューカバー率を管理する場合，レビューカバー率の算出自体が可能であっても，「レビューカバー率がどの段階でどの程度あれば適切にレビューが行われていると判断できるか」という基準値が明確に定められていないならば，プロジェクト管理者は，明確な判断基準が存在しないため利用が難しいと判断し，該当する管理指標を利用しないと考えられる．

二つ目の理由としては，管理者が指標の目的や利用により期待できる効果を把握していないことが挙げられる．このように効能が不明な管理指標は，管理コストを削減するために，利用されないことが多いと思われる．

なお，レビューのように作業担当者の能力に大きく依存する作業は，定量データに基づくプロセス管理を行うよりも，単純に優れた人材を投入することで改善できるというコメントもあった．このような捉え方をされている作業については，組織全体を通して共通で利用できる測定項目を準備し，実態に沿った管理指標を用意すること自体が困難である．また，仮に管理指標が用意されても効果的な利用はあまり期待できないと言える．

プロセス品質保証に関する指標群

表 3.7 より，プロセス品質保証に関する指標を使わなかった理由として，指標運用の対象外であるというコメントが 21 件中 10 件あることがわかる．コメントを詳細に調査したところ，いくつかのプロジェクトでプロセス品質保証部門によるプロセス品質保証を行う対象となっておらず，そのため指標の適用対象外であったことがわかった．また，表 3.5 から指標の目的が不明と回答している管理者や，指標の目的や効果に疑問を感じている管理者が多いこともわかる．加えて，指標に用いられる定量データに疑問を感じている管理者も複数存在する．例えば，指標#36 は，プロセス品質保証作業が適切に実施されているかを監視するため，定量データとして作業の改善指示に対して是正措置をとった件数と，未措置の件数を取得する．しかし，この定量データに対し，「プロセス品質保証の結果指摘された事項は，必ず是正措置をとるため意味がない」といった意見が見られた．

これより，指標定義が組織の実態に即していない場合には，その重要度が十分に理解されない，あるいは利用がそもそも不可能となるために，結果として利用件数が低くなっているものと推察される．

リスク管理に関する指標群

図 3.1 より，指標#38（リスク要因ごとの発生確率）は利用されていない傾向にある．指標#38 の定義によると，この指標の利用には，社内で利用されているチェックシートをもとに，実際に発生した問題件数やプロジェクト件数を集計する必要がある．しかし，コメントを詳細に調査した結果，実際にはこのチェックシートが一部のプロジェクトで使用されていないことがわかった．また，チェックシートを使用していた場合も，プロジェクト件数がチェックシートから直接収集できず，利用が困

難であると指摘している．

このようにこの指標は，利用に必要な定量データの定義がデータの収集源であるチェックシートの運用実態と乖離しているために，データ収集が困難であると判断され，利用が敬遠されていると考えられる．また他にも，指標定義中の「発生確率」という用語の意味がわからないという指摘も見られた．

支援プロセスに関する指標群

支援プロセス（調達管理・構成管理のためのプロセス）に関する指標群に属するもののうち，指標#45（測定項目ごとの測定率）は他の指標と比べても利用率が低い．表 3.5 より，指標の目的や定義が不適切であるという回答が 43 件中 14 件ある．これに関連するコメントについて確認したところ，この指標群に属する指標を用いることで何を把握することができるのかわからないと回答している管理者が複数存在した．また，表 3.7 から，その必要性についても同様に，多くの管理者から疑問が呈されており，そのコメント数は利用しなかった指標に関するコメント 20 件中 9 件に及ぶ．さらに，指標とは異なる別の手段でチェックを行っているプロジェクト管理者や指標定義で用いられている用語に関する理解が不足している管理者がいることが表 3.7 からわかる．実際にコメントを詳細に調査したところ，指標としてデータを収集せず，社内で整備されている別のツールを利用して管理をしているプロジェクト管理者がいた．このような管理者の中には，支援プロセスに関する指標の必要性に理解を示す者もいた．

これより，プロジェクト管理者はこの管理指標群の目的である調達管理・構成管理の重要性は認識していると考えられる．しかし，支援プロセスに関する管理指標は，指標以外の別の手段で十分に管理ができるため必要ないと多くの管理者が考えている．また，指標中で使用されている用語に関する理解が十分でないことも，利用されていない原因と考えられる．このように，既に別の手段で十分に管理が達成されている場合には，新たに指標を利用した定量的管理を行う必要性は低いと言える．

利用が敬遠される指標に関するまとめ

有効に活用されていない指標について精査した結果，管理指標が有効に活用できていないのは，他の指標を活用して利用している以外に，表 3.7 に示すように，「定

義されているデータでは管理できない」「データ収集に手間がかけられない」「指標を利用する必要がない」といった理由が確認された.

以上のヒアリング結果から，我々は対象組織で定義された指標が有効に活用されない理由として，1. 管理指標定義に柔軟性がない，2. 定量データの質にプロジェクトマネージャが問題があると考えた，3. 開発組織において定量データの収集体制が未整備，4. 組織へのフィードバックという観点で管理者の意識が低い，の 4 つがあると考えた.

3.5.4. 管理指標を導入して定量的管理を行う際の指針

本節では，3.5.1 節から 3.5.3 節での考察をふまえた上で，管理指標運用がうまく働いていない場合にどのような点が原因であったかを整理し，改善策についてまとめる.

テーラリングを念頭に置いた管理指標定義にする

組織の実態に合わない収集が行われないようにするには，組織の実態に合わせて管理指標を定義する必要がある．しかし実際には，個々のプロジェクトの状況によって収集が困難なデータが存在するケースも考えられる．実際に，本来はソースコード行数や作成したドキュメント数など，開発規模を表す複数のデータを収集するよう定義されているが，その一部しか測定を行わなかった，という事例をヒアリングから見つけることができた．このような調整が行われている場合，その原因は定量データとして具体的な項目を指定していることにあると考えられる.

よって，管理指標定義を作成する際には，各プロジェクトにおいて調整することを念頭に置いた定義にすれば，組織の実態に合わない収集を回避し，プロジェクトの状況に沿って指標を利用することが可能になる．具体的には，それぞれの管理指標利用に必要な定量データを，個々のデータが表す属性によって分類，整理する．例えば，「ソースコード行数」や「作成したドキュメント数」という定量データは，「開発規模」という属性を表すことができる．あわせて，個々の定量データについて，測定可能な工程やタイミングについても整理する．このような分類，整理を組織レベルで行うことで，管理指標定義で指定されている定量データが収集できない場合でも，プロジェクトの実態に沿って同じ属性を持った代替となる定量データを選択するこ

とが可能となる．

ただ，代替となる定量データを利用する場合，プロジェクト管理者は，代替となるデータ項目を選択し，その収集方法を規定して開発プロセスに取り込む必要がある．すなわち，実際にプロジェクトで管理指標を利用する際に要する手間が増えてしまう．そこで，定量データを置き換える際に，どの指標ではどのような属性を持つ定量データを利用しているのかを理解しやすくしたり，単位系やデータ自体を自動的に変換したりするような仕組みをツールとして運用することや，調整作業に関するガイドラインを策定しておくことが必要であろう．

定量データ収集体制の整備

今回の調査対象となった管理指標群は，先にも述べたように，CMMI ベースによる組織改善を進めるために策定されたという経緯がある．そのため，指標定義中にも CMMI に関する用語が多く含まれている．

一方，ヒアリングの回答者であるプロジェクトマネージャは，PMBOK（Project Management Body of Knowledge）[69] に関する知識は職務上十分に習得している（調査対象組織では PMBOK の知識を習得していることが，プロジェクトマネージャの認定条件となっているため）が，CMMI の用語に関してはプロジェクトマネージャによって理解の度合いにばらつきが見られる．

実際に表 3.7 を見ると，「定義されているデータでは管理できない」という回答がある．コメントを精査すると，例えば，「構成監査」という用語を知らないプロジェクトマネージャは，それに関連した指標を利用していないケースがあった．また，調査対象組織では，CMMI レベル 4 に沿って管理指標を策定している．そのため，実際には組織的な収集体制が整備されていないデータが求められているなど，現状の体制にはそぐわない管理指標も存在する．例えば，レビューに関する指標のように，適切な基準値が組織内で明確に定められていない指標に関しては利用率が低い傾向が確認された．

このように管理指標の中には，指標定義が (少なくとも現状の組織では）適切でない場合が存在する．このような指標群に関しては，社内において定量データの収集体制を整備するとともに，指標を運用する際のガイドラインや用語集，サンプルデータ集などの整備と要員の訓練が必要である．また，指標定義の整備が十分かどうか

を常に確認するために，本研究で行ったようなヒアリングやアンケートを定期的に実施することも重要である．

また，そもそも CMMI では，成熟度レベル 2 に到達するための条件としてプロセス領域「測定と分析（MA: Measurement and Analysis）」のための改善活動に取り組むことが求められている．このプロセス領域は，測定能力を開発・維持することを目的としてデータの収集手順や格納手順，分析手順を明記することを求めている．このような観点からも，定量データ収集体制を整備する必要があると考えられる．

プロジェクトマネージャに対する動機付けの明確化

今回調査対象となった管理指標の中には，指標#38 のようにプロセス資産の整備を目的の一部としたものが含まれている．これらの管理指標は利用が避けられる傾向にあった．プロセス資産の整備は，ソフトウェア開発組織が自らのプロセス資産を蓄積し，今後の開発プロジェクトにプロセス資産を活かすことを目的としており，CMMI でもレベル 4 の到達目標の一つとして挙げられている．

しかし，プロセス資産の整備を目的とした管理指標は，現在担当しているプロジェクトで直接効果が現れるものではなく，次回以降のプロジェクトにおいてはじめて測定の効果が見られるものである．これに対し，プロジェクトマネージャの関心は現在自身が担当しているプロジェクトに限定され，プロセス資産の形成に対する目的意識は必ずしも高くない．このことは，進捗管理や支援プロセスに関する指標について「指標を利用する必要がない」と回答している管理者がいる（表 3.7 参照）ことから推察される．また，ヒアリングの内容を確認するために実施した，追加ヒアリングでも同様のコメントが得られた．これより，このような先行投資的な管理指標は，収集コストとの兼ね合いから利用を避けられる傾向にあると思われる．

このような指標を適切に活用するためには，プロジェクトマネージャに対して，単体のプロジェクトだけにとどまらず，組織へのフィードバックを念頭に置いたプロジェクト管理を行うような動機付けを積極的に行うことが必要である．具体的には，「管理指標定義にそれらを利用した際の将来のメリットや，利用しなかった際のデメリットを明記する」ことが挙げられる．また，「管理者に対して測定したデータに基づくフィードバックを容易に得られる環境を用意することで，データ収集の意義を組織に浸透させる」などの対応策が考えられる．例えば，指標#38 は実際に起こっ

たリスクを分析し，組織のリスク管理のプロセス改善に役立てることを目的とした指標であり，測定データとしてリスク要因ごとの発生件数が定められている．もし組織内で，個々のプロジェクトにおけるリスク要因ごとの発生件数とプロジェクトの成否が共有されていれば，プロジェクト管理者は測定したリスク要因と同様の傾向を持つプロジェクトを探すことができる．過去に実施したプロジェクトの中で同様の傾向を示すプロジェクトがあれば，該当プロジェクトでどのような対処を行ったかを参照できる．また，同様の傾向を示す現在実施中のプロジェクトがあれば，該当プロジェクトの管理者同士で，問題意識を共有し連携して対処していくことができる．このようなフィードバックを得られる環境を整備することで，データ収集の意義をプロジェクト管理者に理解させることができる．

Hall らはソフトウェア開発プロジェクトにおいて測定活動を行う際に考慮すべき項目の一つとして，測定に関する動機付けが必要であることを挙げている [26]．小笠原らは開発要員に対するソフトウェアプロセス改善に関する教育体制を整備した結果，ソフトウェアの計測や CMMI に関する理解が高まった事例を報告している [64]．このことからも，プロジェクトマネージャに対する訓練や収集活動について動機付けを行うことは，定量的管理を組織的に導入する上で必要であると考えられる．

3.6. 妥当性の検証

3.6.1. 調査結果の適用可能性

今回の調査は，国内の大規模なソフトウェア開発組織にて行った．調査対象組織では，CMMI レベル 3 相当のプロセス実施能力を持ち，開発プロジェクトを様々な側面から組織的に測定，分析を行い管理している．そのため，特定のプロジェクトや一部の工程のみを測定する場合には，測定分析のためのコストがかかりすぎる可能性がある．また，調査対象としたプロジェクトは，エンタープライズ系の比較的大規模な開発プロジェクトであり，ウォーターフォールモデルのような，開発の初期段階でプロジェクト管理の全体像を策定するようなプロセスモデルに基づいている．よって開発組織の規模やプロジェクトの業種が異なる場合，今回得られた知見が適用できない可能性がある．一方で，今回の調査対象組織は CMMI レベル 3 相当のソフトウェア開発組織であり，同等の能力を持つ開発組織であれば，今回得られた知見

を適用し，定量的管理指標導入の円滑な導入が期待できる．

今回取得したデータから示した管理指標運用の指針は，あくまでプロジェクト管理における管理指標の活用に関するものとなっている．しかし，個々の組織で定められている管理指標の目的は，現在実施しているプロジェクトの管理のみだけでなく，組織的な監査やガバナンスなども含まれる．組織的な観点から考えると，我々が示した指針により，管理指標の運用がより容易となる一方で，運用する指標や収集される定量データが個々のプロジェクトで大きく異なるなど，組織的には十分統制されていない運用となる可能性がある．そのためにも，今回示した指針を組織的に運用，展開するには，システム監査部門や PMO（プロジェクトマネジメントオフィス）などと連携して，組織全体で統制のとれた展開をするよう注意する必要がある．

3.6.2. 調査結果の有用性

調査対象とした管理指標は，調査対象組織内で独自に定義されたものであり，今回議論した内容は，この組織の指標定義に依存している可能性がある．しかし，調査対象とした管理指標定義は，IPA/SEC が公開している測定項目リストを策定する際に参考にしたという経緯もあり，この測定項目リストと同様の項目が多い．今回示した定量的管理指標導入にあっての指針は，直接的には先に示したように，開発組織のプロセス実施能力がある程度高いことを前提としている．一方，IPA/SEC の測定項目リストは，これから管理指標を利用したプロジェクトの定量的管理を進める開発組織にとっては利用が容易なものである．定量的管理を開発組織に導入する際には，測定項目リストを用いることで導入の初期コストを抑えることができる．この測定項目リストを各組織に導入するにあたっては，組織の状況に合わせて定義を修正する必要があると考えられるが，その際にも今回示された知見，指針に従って導入することで，より効果的な指標運用が期待できる．

ヒアリングの結果得られたサンプル数は 13 件であり，統計的に意味のある議論を行うには十分な数ではない．しかし，今回調査を行ったような，「管理指標をプロジェクトでどのように利用しているか」という観点からの分析は，我々の知る限りなされていない．本研究の貢献は，実際に開発プロジェクトを管理しているプロジェクトマネージャが，組織内で定義されている指標をどのように利用しているかを明らかにした点にある．

3.7. 本章のまとめ

本章では，管理指標を開発プロジェクトに導入して定量的管理を行う際に考慮すべき要件を明らかにすることを目的として，ソフトウェア開発組織を対象に，実プロジェクトにおいて定量的管理がどのように実施されているかを調査した．

調査の結果，調査対象組織においては指標の大部分が多くのプロジェクトで利用されていた．その一方で，プロジェクトによっては有効に活用されていない指標や，調整されて利用される指標が確認できた．調整されている指標に関しては，「定量データの一部を取捨選択して利用する」「定義されている定量データとは異なる，代替となる定量データを利用する」「定量データを測定する頻度・単位を変更する」という 3 種類の調整方法をとっていることが明らかになった．有効に活用されていない指標および調整されて利用されている指標について精査した結果，管理指標が有効に活用できていないのは，他の指標を活用して利用している以外に，表 3.7 に示すように，「定義されているデータでは管理できない」「データ収集に手間がかけられない」「指標を利用する必要がない」といった理由が確認された．

以上のヒアリング結果から，我々は対象組織で定義された指標が有効に活用されない理由として，1. 管理指標定義に柔軟性がない，2. 定量データの質にプロジェクトマネージャが問題があると考えた，3. 開発組織において定量データの収集体制が未整備である，4. 組織へのフィードバックという観点で管理者の意識が低い，という 4 つがあると考えた．そして，これらの指標が有効に活用されない理由を検討し，管理指標を導入して定量的管理を行う際の指針として「柔軟なテーラリングを念頭に置いた管理指標定義にする 」「定量データ収集体制の整備」「プロジェクトマネージャに対する動機付けの明確化」の 3 点を示した．

本調査の結果は，管理指標を用いた定量的管理を実践する際には，ソフトウェア開発組織は組織の実態に即した管理指標を定義する必要があることを示唆している．一方，プロジェクト管理者は，開発プロセスおよび管理指標に対する理解を深めておくことで，より円滑に定量的管理を実施することが可能となる．

第 4 章

定量的管理計画立案フレームワーク

本章では，定量的管理計画を立案するためのフレームワークについて述べる．まず，前章で報告した実態調査の結果を踏まえ，定量的管理における管理内容調整のための視点を整理する．次に，管理計画の立案を，標準開発プロセス定義のオーサリングを行う「オーサリングパート」と，個々のプロジェクトの状況に合わせてプロセス定義を修正する「テーラリングパート」に分割し，それぞれの作業において各アクタがどのように振る舞うべきかを示す．

4.1. 定量的管理を立案するにあたっての視点の検討

第 2 章でも述べたように，定量的管理計画を立案するにあたって，多くの組織では，組織標準の開発プロセス定義や管理指標は明確に定義されている．しかし，それらの組織標準を用いて具体的にどのように立案作業を行えば良いか，その手順は明確には定義されていない．また，プロジェクトの特性を適切に判断した上で管理の内容や程度を調整する必要があるが，その視点も整理されていない．

第 2 章で示した定量的管理の一般的な流れより，定量的管理計画の立案作業は，組織レベルでの作業と個々のプロジェクトレベルでの作業に大別できる．組織レベルでの作業では，プロセスエンジニアによる組織標準プロセス定義などの，組織におけるプロセス資産の管理，構築，修正といった整備が行われる．プロジェクトレベルで

の作業では，組織レベルの作業で整備された組織標準を，プロジェクト計画者がプロセスエンジニアと協力して，個々のプロジェクトの特性に合わせて修正作業（テーラリング作業）を行う．各プロジェクトでの測定，分析作業は，プロジェクトレベルでの作業に作成された管理計画に従って行われる．また，定量的管理計画立案作業は，組織標準のプロセス資産の整備や個々のプロジェクトレベルで新規にプロセスを作成するといったオーサリング作業と，既に存在するプロセス資産をプロジェクトの特性に合わせて調整するテーラリング作業にも分割できる．

さらに前章での考察から，テーラリング時に行うべき調整内容には，管理指標自体の取捨選択と，指標や指標に利用される定量データの調整の 2 つのレベルが存在することが導かれる．さらに，定量データの調整については，**定量データの粒度**および**測定作業の実行頻度**という 2 つの観点が考えられる．ここで，定量データの粒度とは，「開発規模」といった抽象的な測定項目に対する「モジュール数」や「コード行数」といった実際に測定するデータの粒度のことを指す．また測定作業の実行頻度とは，各プロセス中に定量データをどの工程のどの時期に測るかといった測定を行う工程・タイミングのことを指す．

このように，定量的管理計画の立案作業は，組織/プロジェクトといった視点と，オーサリング/テーラリングといった視点の 2 つの異なった視点で捉えることができる．次節では，この 2 つの視点に沿って定量的管理計画の立案作業を整理する．

4.2. オーサリング・テーラリングフレームワーク

本研究で提案する定量的管理計画立案フレームワークを図 4.1 に示す．提案フレームワークでは，対象組織が，組織標準の開発プロセス定義および，図 2.3 に示した測定情報モデルに従って整理された管理指標定義を用意していることを前提としている．

本フレームワークは大きくわけて，「オーサリング・パート」と，「テーラリング・パート」の 2 部により構成される．「オーサリング・パート」では，標準開発プロセス定義と管理指標を効率的に作成したり，改善したりするための指針を示し，「テーラリング・パート」を利用することで，管理指標利用のために必要な測定活動の調整作業を体系的に行うための指針を示す．これらを一貫して運用することにより，定

量的管理計画全体の立案作業を支援する．それぞれのパートで実践すべき内容を以下に示す．

オーサリング・パート

プロセスエンジニアは，まず標準開発プロセス定義と管理指標を構築する．次にそれらがプロジェクトの特性を反映するよう，各種定義の新規作成や修正を行う．これらの作業は，図 4.1 に示すように，前者は組織レベルで行われ，後者はプロジェクトレベルで行われる．

また，組織内で利用される標準開発プロセス定義や管理指標は，長期にわたって利用され，その間に組織の実状に合わせてより適した形に改善される必要がある．そのため組織レベルのオーサリングでは，開発プロセス定義・管理指標の構築を継続的

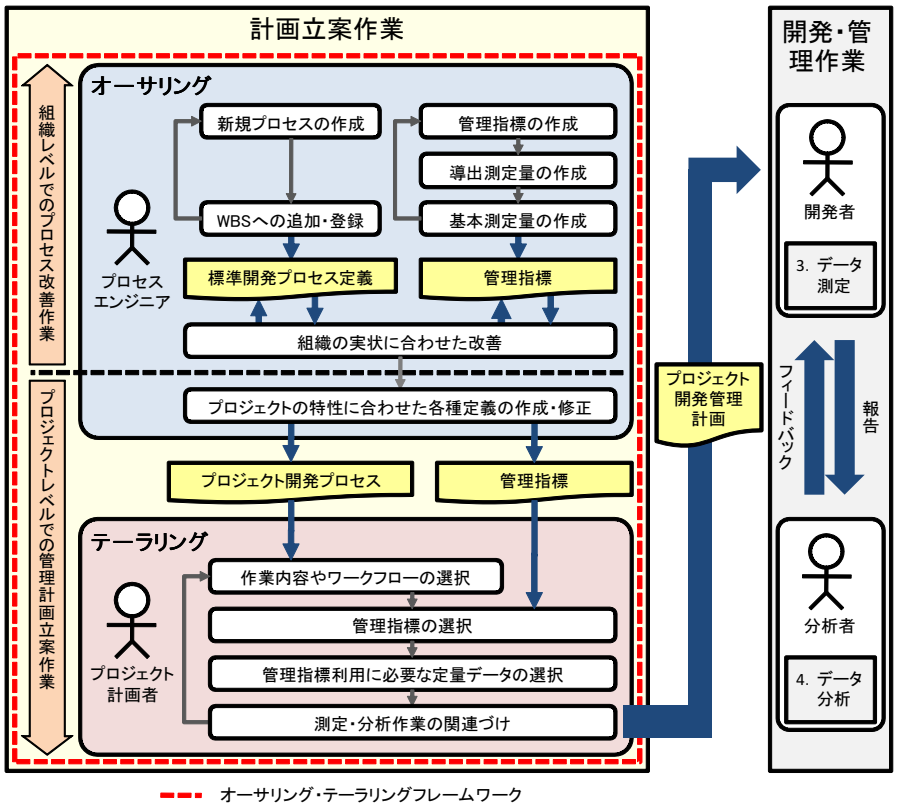

図 4.1　オーサリング・テーラリングフレームワークの概要

に行う．

テーラリング・パート

プロジェクト計画者は，まずオーサリング作業において修正された標準開発プロセス定義をもとに，それらがプロジェクトの特性（予算，人員，納期など）に適合するよう作業内容の取捨選択やワークフローの調整を行う．次に，調整した開発プロセス定義に対して定量的管理計画の組み込みを行う．本組み込み作業では，プロジェクト計画者は，まずプロジェクトの特性に応じて，必要な管理指標を選択する．そして，その管理指標を利用するにあたって必要な定量データを選択し，その測定・分析活動を開発プロセス定義に関連づける．次節では，このようなテーラリング作業の具体例について述べる．

4.3. テーラリング作業の流れ

図 4.2 を用いて，定量データの粒度に関するテーラリングの典型例を説明する．図 4.2 では，今後のスケジュールの変更の必要性を判断することを目的として，基本設

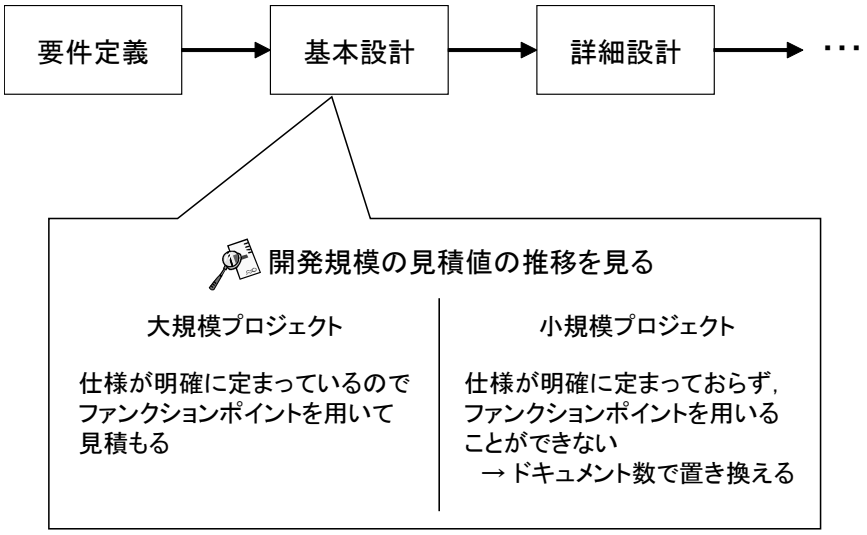

図 4.2　管理指標利用に必要な定量データ選択の例

計の工程で「開発規模の見積値の推移を見る」という管理指標を利用する場面を想定している．ここでは，管理指標の定義で，開発規模としてはファンクションポイントを用いると定められているとする．通常，大規模プロジェクトの場合は，仕様が明確に定まっており，開発規模としてファンクションポイントを用い，見積もりを行うことができる．しかし，小規模なプロジェクトで仕様が明確に定まっていない場合にはファンクションポイントを用いることができない．そこで，計画者は開発規模としてファンクションポイントのかわりに基本設計時に作成したドキュメント数など，代替となる定量データを用いて測定を行う．

一般的にプロジェクト管理者は，担当プロジェクトの管理を目的として複数の管理指標を利用する．管理指標利用にあたっては，プロジェクトの特性（予算，人員，納期など）に合わせて適宜テーラリング作業を行う．このようなテーラリング作業の流れは，図 4.3 のように整理できる．

4.4. 管理計画テーラリングのための支援情報の提供

図 4.3 に示したテーラリング作業の流れは，プロセス構造に対するテーラリングと定量的管理計画の組み込み部分の 2 つのフェーズに分割できる．プロセス構造に対するテーラリングでは，計画者は組織標準開発プロセス定義をもとに，プロジェクトの特性（予算，人員，納期など）に適応させ，変更したプロセス記述を作成する．定量的管理計画の組み込み部分では，計画者は，プロジェクトの特性に応じて，管理指標の中から必要なものを選択する．さらに，その管理指標を利用するにあたって必要な定量データを選択し，その測定・分析活動を開発プロセス定義に組み込む．

本フレームワークでは，特に変更されたプロセスに対する管理計画の組み込みに着目する．変更されたプロセスに対する管理計画の組み込みの際には，指標の選択，および，定量データの収集利用レベルの調整を行う．フレームワークはこの部分について粒度と頻度の視点からの支援情報を用意する．

図 4.4 に，テーラリング時の支援情報の生成と利用に関する概念図を示す．定量データの粒度に関するテーラリングを行う際には，「プロダクト規模」のような抽象データと，「ファンクションポイント値」や「ドキュメント数」など，それに対応した具体的な定量データのセットを定義しておく．これによりプロジェクトの特性に

合わせ，測定するデータを複数の候補から選ぶことができ，柔軟な調整を行うことができる．また，定量データの測定を行う工程，タイミングを細かく指定することにより，全ての定量データに関して測定項目・タイミングが一意に定まり，測定項目の漏れや冗長な定量データの収集活動を未然に防ぐことが可能となる．

つまり，管理指標利用に必要な定量データには，あらかじめ抽象的なデータを割り当てておく．そして，テーラリング時に，管理指標中の抽象的なデータに対応する具体的なデータ形式と測定を行う工程と詳細なタイミングを与えて具体化する．

以下では，第 3 章で調査対象とした実企業の指標群を例に用いて，これらの参考情報の生成方法を示す．

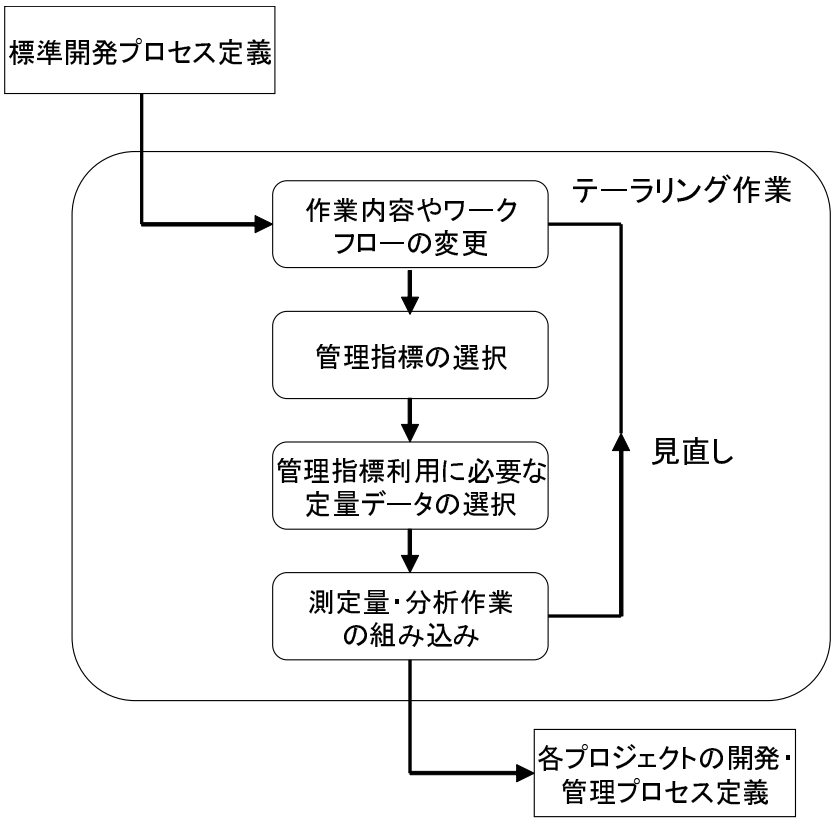

図 4.3　定量的管理計画立案時のテーラリング作業の流れ

4.5. フレームワーク利用に必要な定量データ定義の生成

まず，表 3.1 に示した指標群に存在する全ての基本測定量 190 個（類似・重複も含む）のリストを作成した．このうち，今回は特に「プロダクト規模」という抽象データに対応する具体的な定量データのセットの抽出を行い，プロダクト規模に対応する具体的な定量データの測定が可能な工程・時期に関する分析を行った．

4.5.1. プロダクト規模を表す具体的データ群の整理

先の基本測定量のリスト中で，プロダクト規模を測定する基本測定量のうち，具体的な量（単位）の概念を含むものに着目し，それらを抽出した．詳細な手順を以下に示す．

1. 管理指標の定義の中で「必要な定量データ」として挙げられている項目を抽出する．この作業は 190 個の基本測定量リストを再検討することにより行った．
2. 1. で抽出したデータの中から，プロダクト規模を表すものを抽出する．
3. 2. で抽出したデータの中から，測定単位が不定のものを除外する．

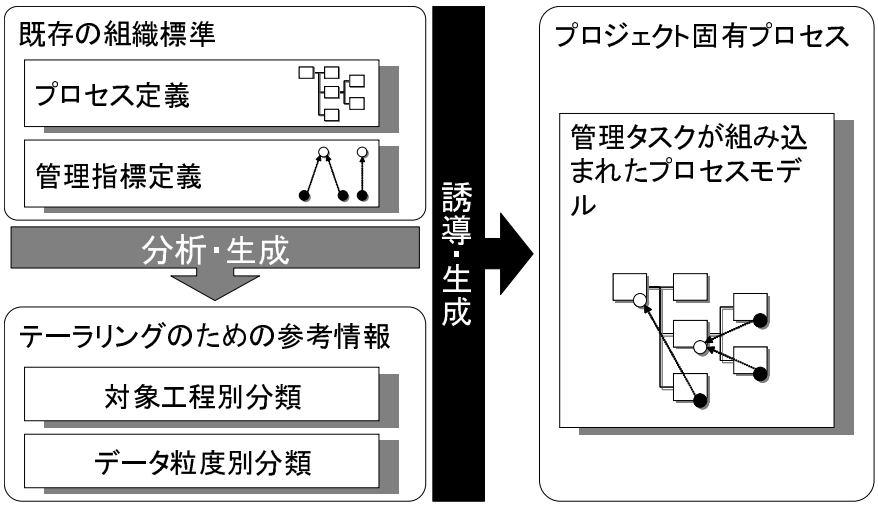

図 4.4　管理計画テーラリング時の支援情報の生成と利用

表 4.1　プロダクト規模を表す具体的な定量データ

名称	単位
DB・ファイル規模	バイト
ドキュメントページ数	ページ数
ドキュメント数	個数
画面・帳票数	個数
コード行数	行数
ステップ数	ステップ
ファンクションポイント	ポイント
プログラム本数	個数

4. 3. で抽出したデータのうち，測定対象工程・時期のみが異なるもの（すなわち名称が同じもの）は同一のものと見なす.

この手順により，表 4.1 に示す 8 つの具体的な定量データ定義のリストを得た.

4.5.2. 定量データの測定時期に関する整理

指標群の元々の定義記述では，測定対象の工程・時期に関して「各工程（完了時）」「プロジェクト完了（後）」といったような対象工程と時期の概念が混在した曖昧な記述が多数含まれていた．測定頻度のテーラリングを行うにあたっては，測定を行う工程・時期をより明確に定義する必要があるため，基本測定量の測定時期や測定頻度の指示形式についても新たに検討を行った．ここでは，先の基本測定量 190 個のリストの中から，対象工程名にタイミングに関する語が含まれているもの，対象タイミング名に工程に関する語が含まれているものをそれぞれ分離・整理した.

整理した結果を表 4.3(a) および表 4.3(b) に示す．表 4.3(a) は対象工程名のみを抜き出して列挙し，対象組織内で特有の工程名称を JIS X0160 に基づいた用語に置き換えを行ったものである．表 4.3(b) は表 4.3(a) で列挙した工程のいずれかに対して測定を実施するタイミングを具体的に指定するための修飾語を列挙したものである．基本的には工程の開始時に計測，終了時に計測，工程中に一定間隔で計測，の 3 種類に整理される.

表 4.2 測定頻度に関する整理の結果

(a) 対象工程

見積
受注
計画
プロジェクト計画書のレビュー
基本設計
機能設計
詳細設計
ソフトウェアコード作成
単体テスト
結合テスト
システムテスト
運用テスト
検証
顧客側の受け入れ検査

(b) 対象時期

開始時
終了時
月毎
週毎

4.5.3. テーラリングのための参考情報

データの測定時期および頻度の指定は表 4.3(a) と表 4.3(b) の直積のサブセットとして表現される．表 4.1 のプロダクト規模を表すデータ名それぞれを計測する時期として可能性のあるものを整理した結果が表 4.3 である．表 4.3 において，各行はプロダクト規模に関する具体的な定量データを表しており，各列は，表 4.3(a) に示した各定量データが測定可能な工程を示している．また，表中の E もしくは I は，各工程の終了時もしくは工程中の一定間隔（週ごと，月ごとなど）に測定可能であることを示している．表 4.3 を利用することで規模に関する定量データの粒度と頻度の調整が容易になる．

表 4.3　プロダクト規模に関する定量データ

	計画	受注	基本設計	機能設計	詳細設計	コード作成
DB・ファイル規模	E	E	I	I	I	I
ドキュメント数	E	E	I			
ドキュメントページ数				I	I	I
画面・帳票数	E	E	I	I	I	I
コード行数				I	I	I
ステップ数	E	E			I	I
ファンクションポイント	E	E				
プログラム本数	E	E	I			

表中の E は対象工程の終了時，I は工程中に一定間隔で測定を表す．

4.6.　提案フレームワークによるテーラリング手順

前提として，先にも述べたように，組織標準の開発プロセス定義および管理指標定義が用意されているとする．また，管理指標定義では，指標利用に必要な項目として「プロダクト規模」といった抽象的な測定項目が記載されているとする．このとき，提案するフレームワークでの管理指標利用に必要な定量データの調整作業は以下の手順で行われる．

1. プロジェクトで利用する管理指標を選択する．
2. 1. で選択した管理指標に定義された抽象的なデータに対応する具体的な定量データを，プロジェクトの特性に合わせ選択する．このとき，対象工程によっては計測できない定量データや，測定可能な工程が限られているものもあるが，これらは表 4.3 によりあらかじめ知ることができる．
3. 2. で選択した具体的な定量データの測定頻度をプロジェクトの特性に合わせ決定する．

図 4.5 に示す，開発期間 3 年，開発人数 1000 人程度のバックエンドシステム開発プロジェクトを例に，上記に示した調整作業の流れを具体的に説明する．今，基本設

計工程において「開発規模の見積値の推移を見る」という管理指標を利用する状況を考える．基本設計工程において採用された，「開発規模の推移を見る」という管理指標で必要となるデータ「開発規模」として用いることのできる具体的な定量データを表 4.3 より選択する．プロダクト規模に該当し，基本設計工程において測定可能な定量データは，「DB・ファイル規模」「ドキュメント数」「画面・帳簿数」「プログラム本数」である．ここでは，DB・ファイル規模をプロダクト規模として採用することとする（図 4.6）．

次に計画者は，測定タイミングを決定する．この例では，DB・ファイル規模を基本設計工程で測定可能なタイミングは，表 4.3 より「週 1 回」や「月 1 回」といったような工程中の一定期間ごととなっている．ここでは，週 1 回測定を行うこととする（図 4.7）．

以上の作業を経て，「開発規模の推移を見る」という管理指標を利用するために必要な測定活動として，「基本設計時に DB・ファイル規模を週ごとに測定する」作業をプロジェクトの開発・管理計画に組み込むことができる（図 4.8）．

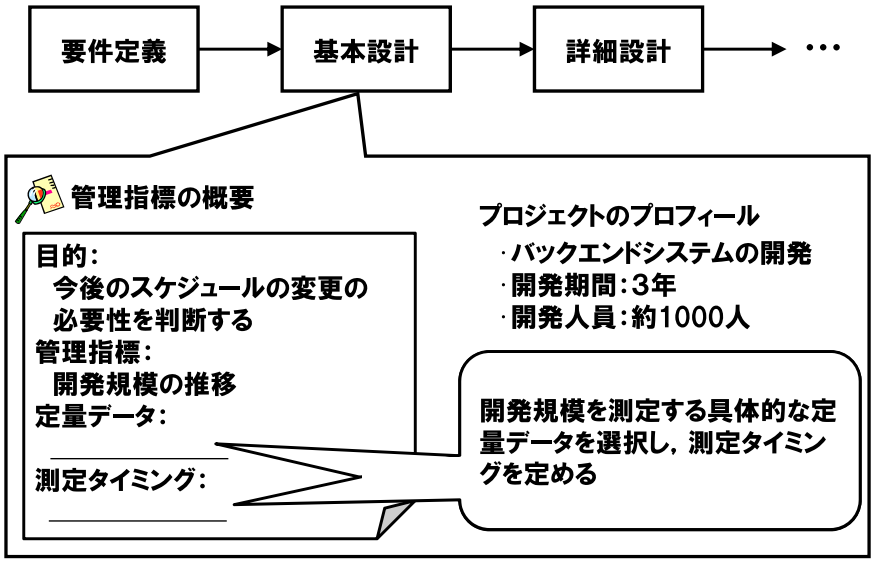

図 4.5　測定項目の調整作業例

4.7. 考察

提案フレームワークは，適用対象組織において開発プロセス定義と管理指標定義が整備されており，かつそれらが組織においてある程度最適化されていることを前提としている．フレームワークを利用したテーラリングは，開発プロジェクトの初期段階で全工程の管理プロセスがテーラリングされることを想定している．よって，プロジェクトの初期段階において，開発プロセスとあわせてその管理プロセスが策定されるような体制である必要がある．

本フレームワークでは，テーラリング時に管理指標を利用するのに必要な定量データに関する参考情報を利用する．この参考情報は本フレームワークで導入したものである．本研究では，この参考情報を作成するにあたって，定量データを整理する視点として，データの粒度と測定のタイミングと言う 2 つの視点を示した．本フレームワークでは，開発計画の立案にあたって必要なコンテンツ（開発プロセス定義，管理指標定義，テーラリングのための参考情報）と，それを利用したオーサリング，テー

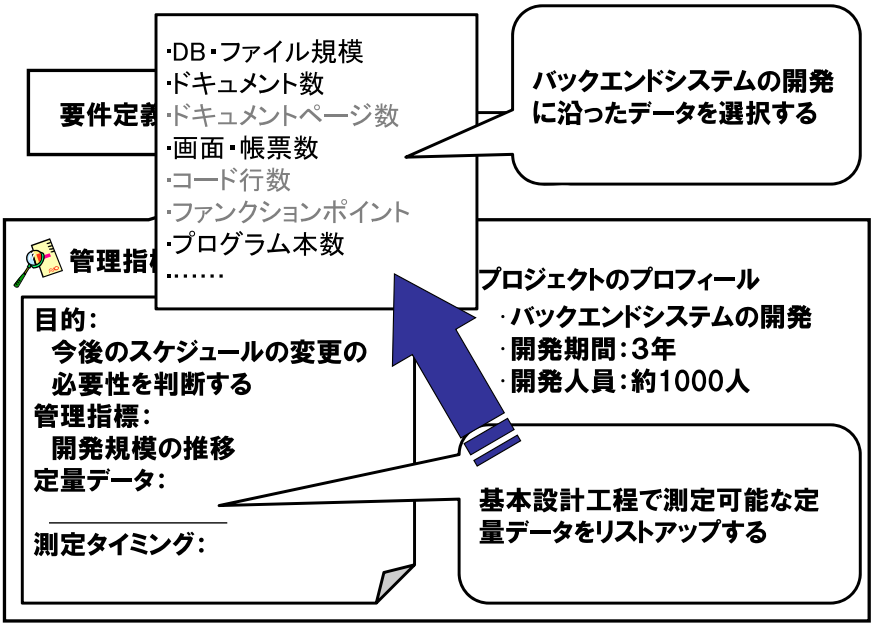

図 4.6　定量データの選択

ラリングの手順を定めている．一方で，これらフレームワークが必要とするコンテンツが具体的にどのようなものであるかは，組織によって異なる．これは，組織標準として定められる開発プロセス定義や管理指標，およびその利用に必要な定量データは組織によって異なるため，組織の状況に合わせた調整が必要となるためである．

本フレームワークの貢献は，第 3 章で実施した実態調査に基づき，これまでプロジェクト計画者の経験に基づいて行われていた計画立案作業を体系的にまとめた点にある．また，テーラリングに関する参考情報を導入することで，プロジェクトの特性に合わせて定量的な管理計画を柔軟に立案することが可能となる．ただし，具体的な定量データは，測定可能な工程や収集間隔がそれぞれ異なるために，具体的な定量データを選択する作業は複雑となる．そこで，次章では，フレームワークを用いた管理計画の立案作業を支援するシステムについて述べる．

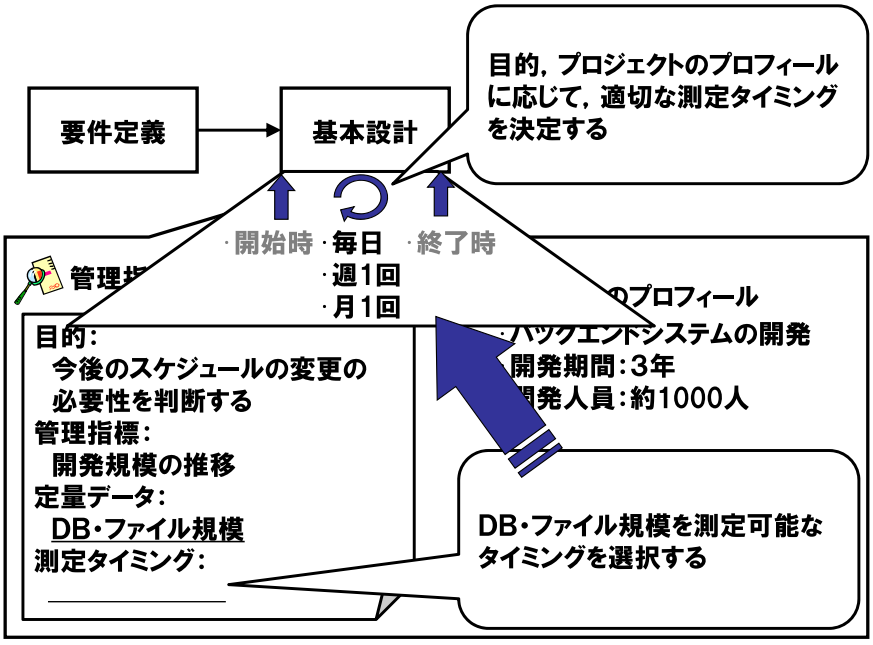

図 4.7　測定タイミングの選択

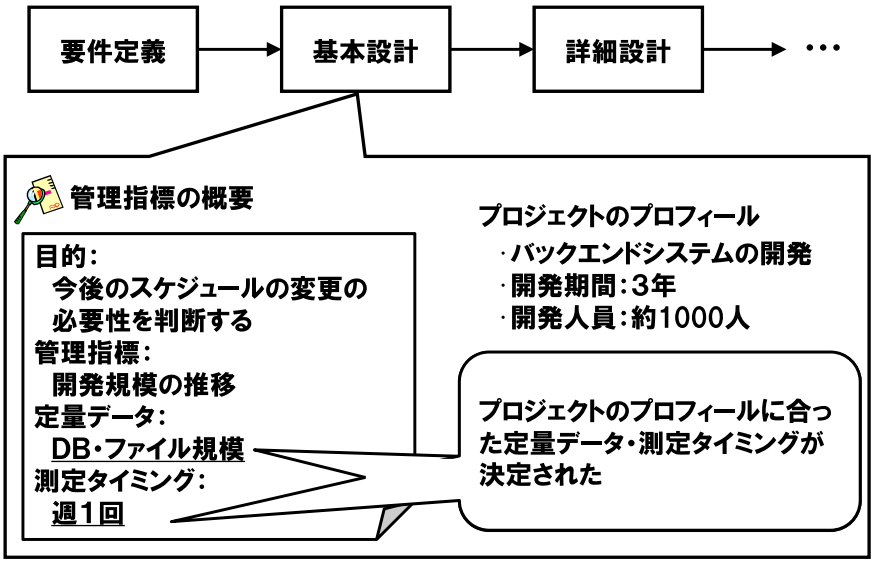
要件定義
基本設計
詳細設計
・・・
管理指標の概要
目的:
今後のスケジュールの変更の
必要性を判断する
管理指標:
開発規模の推移
定量データ:
DB・ファイル規模
測定タイミング:
週1回
プロジェクトのプロフィール
・バックエンドシステムの開発
・開発期間:3年
・開発人員:約1000人
プロジェクトのプロフィールに合った定量データ・測定タイミングが決定された

図 4.8　測定項目の決定

第 5 章

定量的管理計画立案支援システムの構築

本章では，前章で提案したフレームワークに基づく定量的管理計画立案を支援するシステム AQUAMarine の開発について述べる．AQUAMarine では，提案フレームワークに基づいて，プロセスエンジニアが行う標準開発プロセスのオーサリング作業, および, プロジェクト計画者の行うテーラリング作業を支援する．AQUAMarine は電子プロセスガイドとしての機能もあり，測定者が定量データの測定を行う際に，プロジェクトで選定された測定項目について，その収集方法などを参照することができる．本章では，AQUAMarine の構成について述べるとともに，実開発企業のプロジェクト管理者による，AQUAMarine のレビュー結果についてもあわせて報告する．

5.1. AQUAMarine の設計と実装

5.1.1. システムの概要

AQUAMarine は，第 4 章で示したフレームワークに従い，定量的管理プロセスのオーサリング・テーラリング作業を支援する．プロセスエンジニアに対しては，標準開発プロセス定義と管理指標の新規作成作業やプロジェクトの特性に合わせたこれらの修正作業を支援する．また，プロジェクト計画者に対しては，管理指標やその

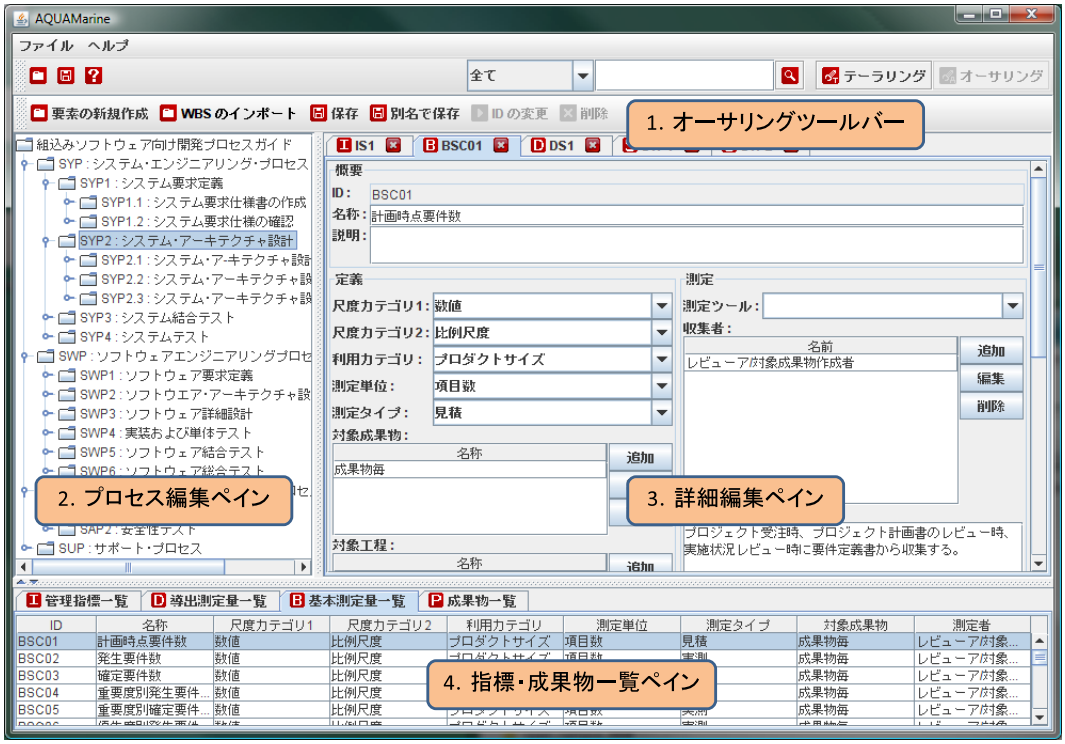

図 5.1　AQAUAMarine のスクリーンショット（オーサリングモード）

利用に必要な定量データの取捨選択作業，開発作業と定量データの収集作業との関連づけを支援する．さらに，測定者・分析者に対しては，定量データの収集・分析方法の理解を促進する機能を備える．AQUAMarine は標準開発プロセス定義と管理指標を入力とし，それらに基づいて実プロジェクトのプロセスのオーサリングおよびテーラリングを支援する．作業結果は，定量的管理プロセスが統合された開発管理計画として出力される．

5.1.2.　システムの構成と提供する機能

AQUAMarine は Java で実装され，入力として標準開発プロセス定義と管理指標が記述された XML 形式のプロジェクトファイルもしくは CSV 形式で記述された開発プロセス定義を読み込んで機能する．

AQUAMarine のスクリーンショットを図 5.1 および図 5.2 に示す．以下では，シ

図 5.2　AQAUAMarine のスクリーンショット（テーラリングモード）

ステムの機能について説明する．特に，本論文では，システム全体の機能のうち，フレームワークの実現に直接関連のあるオーサリング機能群（図 5.1）とテーラリング機能群（図 5.2）の 2 つを説明する．

オーサリング機能群

A．各種定義の新規作成支援機能

WBS 形式で表された開発プロセスの新規定義，新たな管理指標の追加といった，各種定義の構築作業をウィザード形式で支援する．開発プロセスの構築においては，単一の開発作業を表した「タスク」と，その「タスク」の集合である「フェーズ」を階層的に定義・記述することができる．また，開発プロセスと関連する「成果物」と CMMI を考慮したプロセス構築を支援するための「プロセスエリア」を定義・記述することができる．測定量に関連する要素としては，ISO/IEC 15939 規格で定められている測定情報モデルをもとに，プロジェクトにおいて実際に測定可能な属性で

ある「基本測定量（定量データ）」，複数の基本測定量をもとに算出される「導出測定量」，プロセスの状況を表現した「管理指標」の定義・記述が可能である．また，これらの定義には，具体的なドキュメントのサンプルや収集方法などに関する情報を含めることもできる．

これらの機能により，開発組織の資産として，開発プロセスや管理指標の定義を管理・蓄積することが容易となる．また，具体的なサンプルをあわせて蓄積することができ，これまでのプロジェクトを通して得られた実践的な例を蓄積していくことが可能となる．

B. 各種定義の修正支援機能

プロジェクトで用いる各種定義の改善・修正作業を支援する機能を提供する．開発プロセス定義や管理指標は，大量の設定・記述項目を含むため，それぞれの項目をグループ化し，改善・修正しやすいようにしている．また，開発プロセス定義・管理指標と関連する成果物・プロセスエリアの情報も含め，それらを直感的に修正できるよう支援する．さらに，各管理指標で用いる定量データに関する情報を編集できる機能を有す．具体的には，定量データが測定可能な対象工程・測定タイミングを編集できる．また，プロジェクトの状況に応じて，より適当な定量データを利用できるようにするために，代替となる定量データを定義することができる．

この機能により，組織の実情に合わせたプロセス資産の修正を容易に行うことができる．また，定量データの測定可能な対象工程や代替となるデータなど，定量データの調整方法に関する知見を蓄積していくことができる．

テーラリング機能群

C. 定量データの測定・分析活動設定機能

定量データの測定分析活動を行うには，個々の定量データを測定する工程とその工程に合った測定タイミングを指定する必要がある．また，場合によっては，測定対象の定量データが対象工程において得られずに，その代替となる定量データを選択する必要が生じることがある．システムは，定量データごとにその測定可能な対象工程，測定タイミングと，代替となり得る定量データに関する記録を保持している．システムは，この情報をもとに，測定対象工程に合った測定タイミングの表示と，関

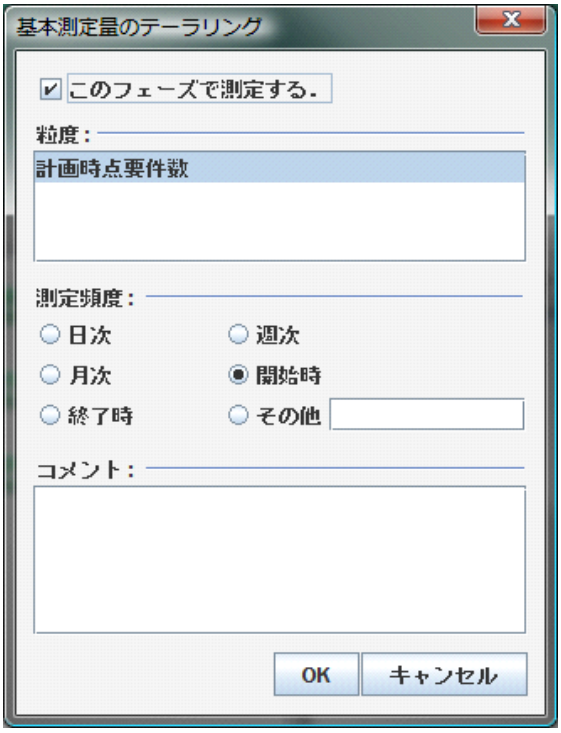

図 5.3　基本測定量の調整ダイアログ

連する代替定量データのリストアップを行う（図 5.3）.

これにより，過去に行われたテーラリング作業で得られた知見を利用して，開発作業への測定・分析活動の設定を柔軟に行うことができる.

D. 開発管理計画の閲覧機能

定量データの測定分析活動が統合された開発管理計画は，その量も膨大なものとなり，計画全体を確認することが困難となる．本機能は，開発プロセスとそれに関連づけられた定量データをダイアグラムとして画面上に並べて表示することで（図 5.2 中の 6．確認ペイン），開発プロセスと定量データの関係を直感的に把握できるよう支援する．また，開発プロセスや定量データに関して，具体的な作業内容や測定方法を表示する機能も有している（図 5.2 中の 7．詳細表示ペイン）.

このように，各工程やその作業で必要となる測定作業が図示されることで，具体的なイメージを持ってプロジェクトに臨むことができる．また，管理指標や定量デー

タに関する詳細な情報を容易に参照できるので，指標や定量データに関する理解が促進されると考えられる．

5.1.3. システムの利用シナリオと提供する機能

AQUAMarine システムの活用法をわかりやすく示すために，プロセスエンジニアとプロジェクト計画者の視点から，AQUAMarine を用いた定量的管理計画立案作業のシナリオを以下に示す．このシナリオではプロセスエンジニアが標準開発プロセス定義と管理指標を新規にオーサリングし，それらをもとにプロジェクト管理者がプロジェクトごとのテーラリングを行うことで，最終的な開発・管理計画を出力するという作業の流れを想定している．

1. プロセスエンジニアは，システムを実行し，標準開発プロセス定義と管理指標を新規に構築する．そのために各種定義の新規作成を支援する．
2. プロセスエンジニアは既存の標準開発プロセス定義と管理指標を改善する．システムは各種定義の修正支援機能を提供し，改善活動を支援する．
3. プロセスエンジニアおよびプロジェクト管理者はプロジェクトに合わせて標準開発プロセス定義と管理指標を修正する．システムは各種定義の修正支援機能を提供し，プロジェクトに合わせた修正を支援する．
4. プロジェクト管理者は，管理指標の中からプロジェクトに必要と考えられるものを選択し，その利用に必要な定量データを選択する．システムは，管理指標とそれに関連する定量データ間の依存関係や構造を直感的に把握できるよう支援する．
5. 4. で選択した定量データを測定する工程・頻度を決定する．この時システムは，選択した定量データについて，測定可能な工程や頻度をリストアップするなどの支援をする．
6. プロジェクト管理者は計画全体を確認する．必要であれば手順 4 に戻り，再度管理指標・定量データを選択する．システムは測定・分析活動と開発プロセスの関連を直感的に把握するために必要な支援を行う．
7. 選択した管理指標の全てに関して測定活動の組み込みが完了したら，測定・分析活動を統合した開発・管理計画を出力する．

5.2. AQUAMarine の評価

5.2.1. 実施概要

本節では AQUAMarine の有効性を確認するために実施したシステムの評価について述べる．

評価では，あらかじめ AQUAMarine の評価項目と簡単な利用方法を整理したレビューシートを作成し，実企業のプロジェクト管理者らに配布した．そして，レビューシートに基づいて AQUAMarine を利用・評価してもらうことで，システムの有効性について回答結果を収集した．評価に協力していただいたのは，第 3 章の実態調査を実施したソフトウェア開発組織である．また，回答者は調査対象組織において，定量的管理を実施しているプロジェクト管理者，もしくは定量的管理を行ったことのあるその他の管理者を対象とした．

評価にあたっては，AQUAMarine 本体に加え，AQUAMarine の詳細な利用手順を掲載したユーザーズガイド，協力組織内部で利用されている標準プロセス定義・管理指標を電子化したプロジェクトファイルを協力企業に配布した．

5.2.2. レビューシートの概要

レビューシートでは，評価全体を大きく 3 つに分類しており，それぞれの分類は機能・目的ごとにまとめられた複数の質問項目から構成されている．回答者はそれぞれの質問項目に応じて，6 段階（未回答，そう思わない，あまりそう思わない，どちらとも言えない，ややそう思う，そう思う）もしくは，自由記述による評価を行う．3 つの分類と各分類における質問内容を以下に示す．

A) オーサリングにおける支援機能の有効性とユーザビリティ

開発プロセスのドラッグアンドドロップによる並び替え機能，各種定義の編集機能・新規作成支援機能について，その有効性とユーザビリティを評価した．

B) テーラリングにおける支援機能の有効性とユーザビリティ

計画の閲覧機能，測定量（定量データ）の関連づけ機能・測定タイミング設定機能について，有効性とユーザビリティを評価した．

C) システム全体の有効性

システムの目的が理解しやすいか，定量的管理計画の立案を有効に支援しているか等について評価した．

5.2.3. 評価結果

本評価では，最終的に 11 件の回答を得た．回答者の属性としては，経験年数が 5 年から 22 年で平均が 12.2 年，プロジェクト経験回数が 0 回から 15 回で平均は 6.2 回であった．

評価結果を図 5.4 から図 5.6 に示す．各図において，縦軸は機能や目的ごとにまとめられた質問項目を示しており，横軸はそれぞれの質問項目に対する 6 段階の回答を段階ごとに積み上げたものである．

5.2.4. 評価対象ごとの傾向分析

ここでは，観点ごとにどのような傾向が見られたかを述べる．

A) オーサリングにおける支援機能の有効性とユーザビリティ（図 5.4）

図 5.4 からは，新規作成機能において大量の要素を作成しづらいという点を除き，オーサリングに関する機能群およびインタフェースが利用者にとって利用しやすいということが読み取れる．評価の低い新規作成機能に関しては，「ウィザード形式なので，大量の要素をまとめて作成するのには向いていない」という指摘があった．

B) テーラリングにおける支援機能の有効性とユーザビリティ（図 5.5）

図 5.5 からは，テーラリング機能およびインタフェースのユーザビリティが比較的高いことを読み取れる．ただし，閲覧機能の表示方法が直感的でないことや，ダ

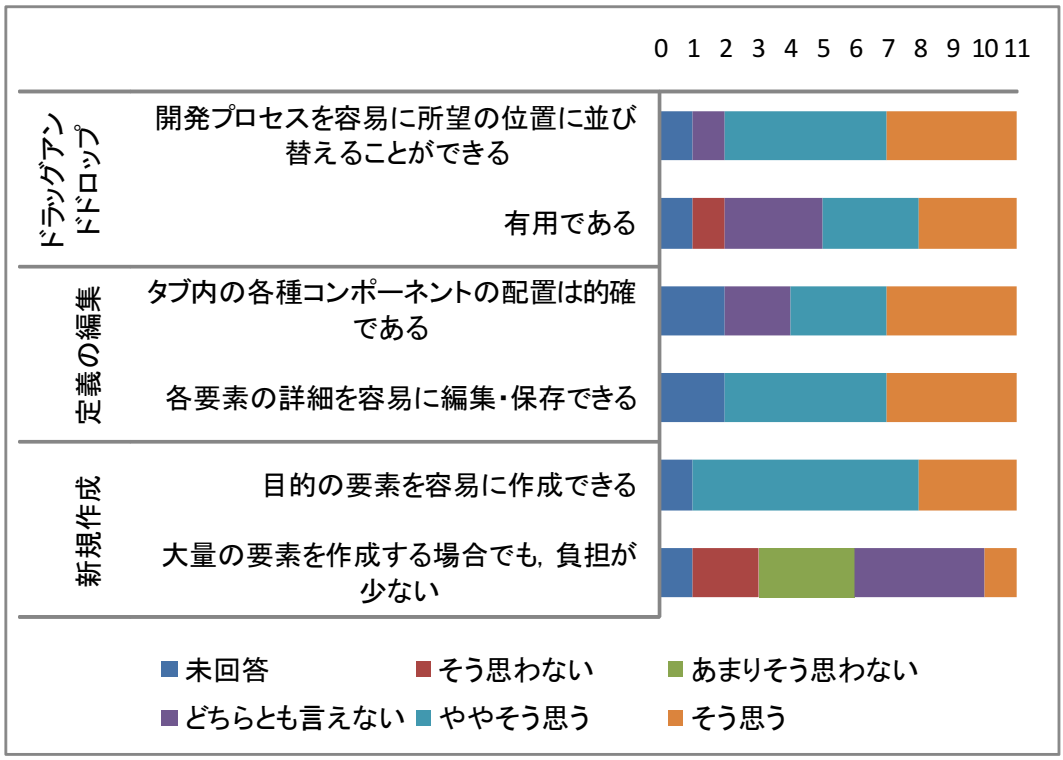

図 5.4　オーサリング機能に対する評価

イアログから測定量を関連づけにくいことがうかがえる．これらの機能に関してコメントを分析したところ，閲覧機能については「プロセスペインを選択して，さらに確認ペインを選択しなければならないのは煩雑」，「画面解像度が低い環境では詳細表示ペインが狭く感じる」といった指摘が見られた．ダイアログから測定量を関連づける機能については，「測定量を関連づけるのに必要なクリック数が多く煩雑である」との指摘を得た．

C) システム全体の有効性 (図 5.6)

システムの機能に関する総合的な有効性や，システムが持つ目的の理解度に関して評価を行った結果を図 5.6 に示す．結果からは，システムが定量的管理計画の立案手順を理解する上で比較的有用であることが読み取れる．ただし，システムの利用方法を理解しにくいという評価も見られた．これは，システムに利用者を誘導するような

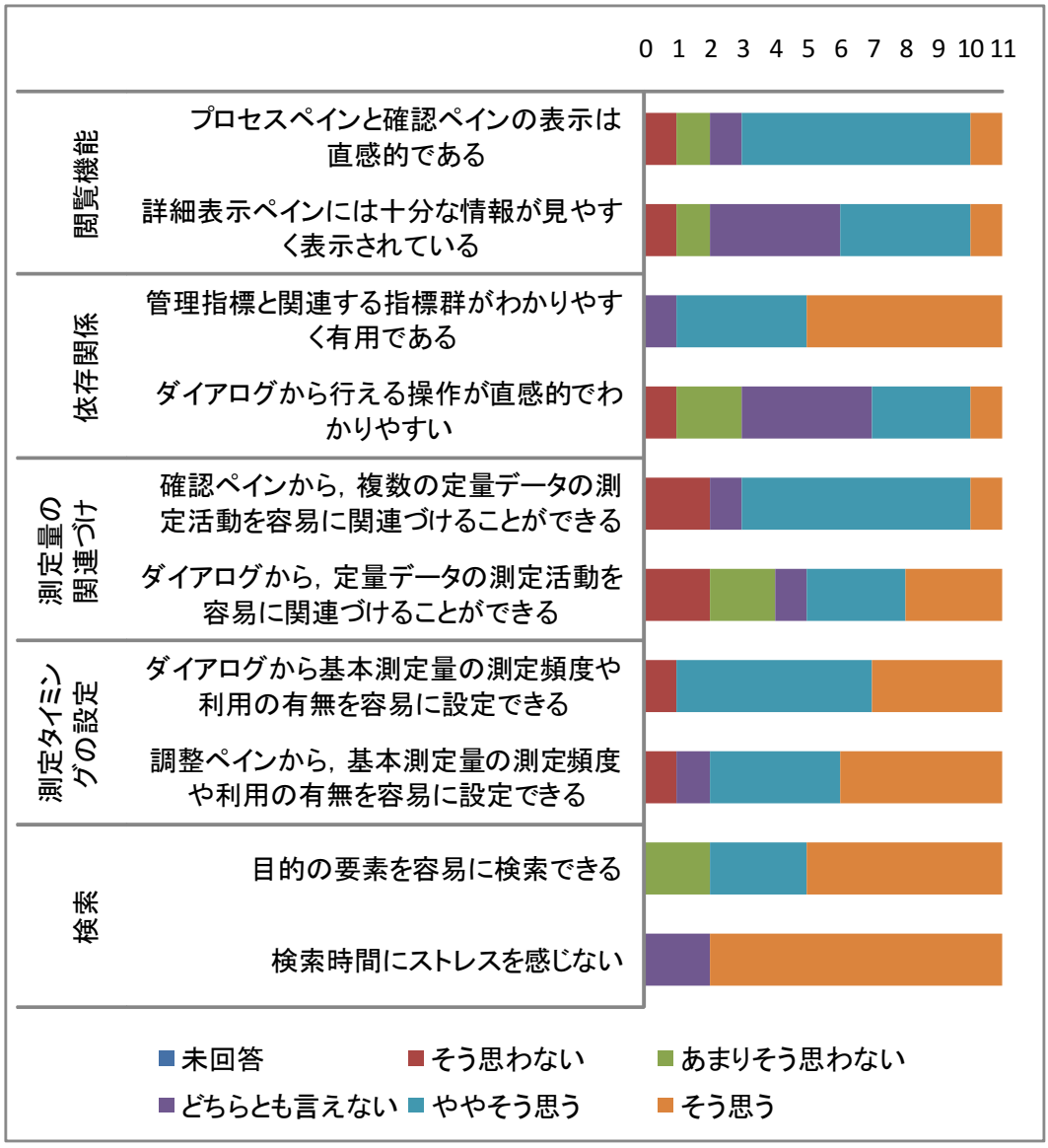

図 5.5　テーラリング機能に対する評価

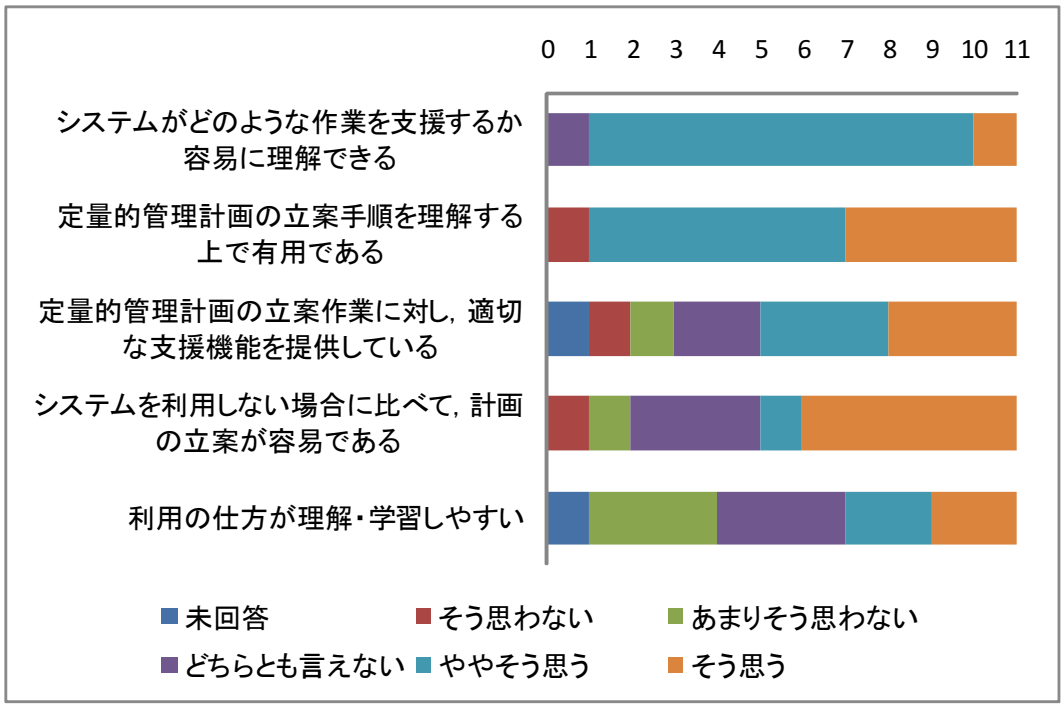

図 5.6　システム全体に対する評価

機能が無いこと，ユーザーズガイドの記述が不十分だったことが理由と考えられる．

システム全体に対するコメントでは，「意図される作業に誘導するような表示が出るとわかりやすい」，「ナビゲーション機能があると便利である」など，システムをより容易に利用するための仕組みが必要であるとの指摘を受けた．また，「情報をエクスポートして加工できる機能があると，システムを利用していないところへも報告できる」，「定量データの測定結果を入力できると便利である」など，立案した定量的管理計画の共有と実践のための機能が必要であるとコメントを得た．さらに，「システムに認証機能を追加し，標準定義を修正できる人物，閲覧のみ可能な人物など，利用者の権限に応じて利用できる機能が制限される仕組みが必要である」との指摘も得た．

5.3. 考察

AQUAMarine は第 4 章で提案したフレームワークに基づき，標準プロセス定義および管理指標定義の作成（オーサリング）作業と調整（テーラリング）作業を支援する．テーラリング作業では，ユーザが選択した管理指標について，その利用に必要な定量データの開発作業への関連づけを行う．また，計画立案作業や収集作業の支援を目的として，開発作業項目や測定項目に関して詳細な情報を参照することができる．

前節で示した結果より，システムは開発管理プロセスのオーサリング・テーラリングに有効な支援機能を提供していると思われる．特に，定量的管理計画の立案手順を理解する上で有用であるとの評価が強い．このことから，AQUAMarine は組織的に統一された形での計画立案を支援すると考えられる．また，チュートリアル（ユーザーズガイド）とナビゲーションを充実させることにより，今後，管理者教育への活用が期待される．

システムに求められる機能としては，情報をエクスポートして AQUAMarine を利用していない関係者からも定量的管理計画を容易に参照できる機能や他のシステムと連携する機能など，立案した計画の実施を支援する機能が求められている．また，システムのユーザビリティに関して多数の改善点を指摘されている．このことから，今後，新機能の追加に加え，システムのユーザインタフェースを見直す必要があると考えられる．

AQUAMarine の利用にあたっては，第 4 章で述べたフレームワークと同じく，開発組織にて標準プロセス定義および管理指標定義が整備されていることが前提となっている．また，AQUAMarine でこのようなプロセス資産を取り扱うためには，所定の形式で定義を作成する必要がある．この点について，AQUAMarine では，多くの開発組織で採用されている表形式のファイルフォーマットの一つである CSV（カンマ区切り形式）ファイルの入力に対応している．よって，既存の形式を AQUAMarine で扱える形式への変換は比較的容易に行える．

AQUAMarine が提供する機能は，計画立案作業の支援と，データ収集のために必要な項目の閲覧機能である．一方で，定量的管理を実施するにあたっては，計画に基づいたデータの収集，収集されたデータの分析，評価が重要となってくる．

AQUAMarine はあくまで計画立案支援を目的としており，このようなデータ収集機能は持たない．また，指標による管理を行う上で重要な基準値の入力や収集した値の解釈は，プロジェクト管理者に委ねられている．

AQUAMarine の貢献は，第 4 章で提案したフレームワークに基づいた管理計画の立案作業を支援することで，プロジェクト計画者に対して体系的な立案作業を容易に行うことのできる環境を提供した点にある．これにより，プロジェクト計画者は，自身のプロジェクトの状況に合わせて柔軟に開発計画を作成することが可能となる．また，AQUAMarine で作成された計画を蓄積することで，プロセス資産の整備にもつなげていくことが可能となる．

5.4. 本章のまとめ

本章では，提案フレームワークに基づき，プロセスのオーサリングとテーラリングを支援するシステム AQUAMarine の開発を行った．本システムは，標準開発プロセス定義や管理指標の新規構築に関わる作業を支援する．また，定量データと利用できる工程の関係のような，これまで明らかになってこなかった，経験に依る知識を蓄積するための仕組みを備えている．プロジェクト計画者に対しては，プロセスエンジニアによって作成された定量データに関する情報を利用して，プロジェクトの特性に合わせた修正，開発プロセスへの測定・分析活動の統合を行うことができる．測定者・分析者に対しては，定量データの測定・分析方法や実例を提示することで，測定・分析に対する理解を促進する．このように，AQUAMarine はプロセス定義や管理指標の構築から，プロセスへの測定・分析活動の統合を一体的に行うことが可能となる．また，AQUAMarine を利用して作成された開発管理計画を蓄積していくことができる．これらの情報から過去の知見を活かしてテーラリングをしていくことも可能となる．

さらに，AQUAMarine の有効性を確認するため，実企業における評価も行った．結果として，提案フレームワークおよびシステムが定量的管理計画の立案作業に対し有効な支援機能を提供していることを確認した．

現在，協力組織において AQUAMarine の導入を進めている．また，実プロジェクトにおける AQUAMarine の適用実験の準備も進めている．今後の展望としては，

システムの利用による長期的な生産性の評価など，実開発プロジェクトにおけるシステムの有用性・妥当性に関する定量的な評価が挙げられるまた，適用実験で得られたテーラリング結果を，組織のプロセス資産として有効活用する手段も挙げられる．

第 6 章

開発記録を利用したソフトウェア開発プロセス分析手法

第 4 章，および第 5 章では，ソフトウェア開発における定量的管理計画立案に関する課題に対して，立案作業のためのフレームワークおよび立案支援システムの開発について述べた．本章，および続く第 7 章では，開発プロセスの定量的な評価，分析に関する課題に対して，開発作業実施時に記録される情報を利用した評価，分析手法について述べる．本章では，開発作業記録の自動収集を念頭においた，ソフトウェア開発プロセスの定量的な分析手法について述べる．

6.1. はじめに

高品質なソフトウェアを生産するにあたっては，ソフトウェアの品質を評価する手法が必要となる．ソフトウェアの品質評価手法は，ソフトウェア作成後にソフトウェアそのものの品質を評価するプロダクト評価と，ソフトウェア開発中にソフトウェア開発プロセスを評価するプロセス評価に大別される．ソフトウェア開発においては，プロダクト評価を行うとともに，プロセス評価を行うことが重要である．プロセス評価は，ソフトウェアの作成中に随時実施でき，早期に問題を検出することを可能にするからである．

このようなプロセスを評価する取り組みの多くは，CMMI のような組織のプロセス実施能力を定性的に評価するものか，成果物の管理情報として記録されたデータ

からプロセスに関する情報を取り出し，作業効率や生産性についての定量的な評価を行うものである．このような研究において，ソフトウェア開発プロセスの品質が，その成果物の品質にどのような影響を与えるか，また，開発プロセスのどのような要因が成果物の品質に影響を及ぼすかは十分に明らかになっていない．

また，ソフトウェアのリリースサイクルが年々短くなっている昨今では，このような開発プロセスの評価・分析を迅速に低コストで行いたい．そのためにも，開発時に自動的に収集されたデータを用いて開発プロセスを自動的に評価するようにしたい．このようにすることで，コストをかけずに開発プロセス中の問題があった部分を迅速に検出し，対応を行うことができる．

本章では，ソフトウェア開発中に自動的に収集される開発作業記録を用いた，ソフトウェア開発プロセスの分析手法について述べる．本提案手法はプロセスデータを手動で収集する必要がないため，プロセスの評価・分析にかかるコストを抑えることができる．また，提案手法による分析結果と成果物の品質との比較評価を行った．本実験により，開発プロセスのどのような要因が成果物の品質に影響を及ぼしたかを明らかにする．

まず，6.2 節で提案手法が対象とするソフトウェアプロセスを形式的に定義し，プロセス分析手法の概要について述べる．次に，6.3 節では保守工程におけるバグ管理システムを利用したバグ修正プロセスを題材に，その作業内容を整理する．6.4 節では，オープンソースソフトウェア開発プロジェクトの一つである Eclipse プロジェクトに提案手法を適用した際の分析例を示す．

6.2. 開発作業記録を利用したソフトウェア開発プロセス分析

6.2.1. 開発プロセスの定義

本研究では，ソフトウェアプロセスを「イベントの発生順序を定義したもの」として定義する．以下では，イベントおよびソフトウェアプロセスの形式的な定義を与える．

イベント

イベントとは，ソフトウェア開発が実施されている際に観測される事象を指す．イベントはソフトウェア開発中に時系列に発生する．今，ソフトウェア開発中に発生したイベントの全集合を E とする．このとき，個々のイベント $e \in E$ は，イベントの発生時刻 t，イベントの種類 k，イベントの属性集合 A，の 3 つの要素からなる．イベントの属性の数 $|A|$ は，イベントの種類に依存する．

イベントの具体例としては，会議における発言や，ソースコードの作成作業，テストの報告など，その種類や粒度は様々なものが考えられる．ここでは，このようなイベントは，ソースコード管理システムやバグ管理システムなどの，開発支援ツールにより自動的な収集が可能なものを想定している．

イベント e は (t, k, A) で記述される．例えば，「2009/3/15 16:31:25 に テスター Alice が，FinderDialog.cpp に関するバグを報告した」というイベントは，下記のように記述される．

```
(2009/3/15 16:31:25, バグ管理, {報告者="Alice", 対象モジュール
="FinderDialog.cpp", バグの状態="報告"})
```

このようなイベント系列は，成果物の管理ツールや変更管理，不具合情報管理ツールの履歴情報から抽出できる．また，EPM などのツールを導入すれば，複数の管理ツールが扱う履歴情報を統一的に自動収集することが可能となる．

ソフトウェアプロセスモデル

ソフトウェアプロセスは，狭義にはソフトウェア開発における作業間の順序関係や包含関係などを意味する．また，これらに加え，対象成果物や開発組織の構造，開発や管理のための方法論までを含めてソフトウェアプロセスと呼ぶ場合もある．このようなソフトウェアプロセスを一定の記法に基づき記述したものをソフトウェアプロセスモデルと呼ぶ．本研究では，狭義の意味によるプロセスに着目する．すなわち，プロセスの持つ振る舞い的な側面を記述したものをプロセスモデルとする．先程のイベントの定義を用いてプロセスモデルを定義すると，プロセスモデルとは「イベントの種類に基づき，イベントの発生順序を定義したもの」とできる．

プロセスモデルの記述方法としては，WBS 形式に基づいた表や文章などが一般的に用いられている．他の記述方法としては，プログラミング記述言語によるものや SPEM，BPMN，UML のアクティビティ図などがある．また，先に示したように，プロセスモデルはイベントの発生順序を定義したものと考えることができる．このようなイベントの発生順序を表現できる記法としては，状態遷移図やペトリネットなどがある．

本研究では，特定のプロセスモデル記述言語を前提とはしない．以降では，便宜的に状態遷移図を用いてプロセスモデルを示す．

6.2.2.　作業記録からの開発プロセスの抽出

バグ管理システム等に記録された開発作業記録をプロセスモデルで定義された作業群に対応付けすることで，プロセスモデルに基づいた解釈が可能となる．具体的には，作業記録中の 1 イベント，1 イベント中の属性情報をプロセスモデル中の 1 作業，もしくは 1 作業の開始または終了に対応付けを行う．

図 6.1 は単純な階層関係を持つプロセスモデルを用いて作業記録の解釈を行う例である．作業 A は作業 B および作業 C の逐次実行に分解されている．この場合，各イベントについて，「作業 A の開始」「作業 B の実行」「作業 C の実行」「作業 A の終了」の 4 種類に対応付けた解釈を行っている．このように，プロセスモデルに対して実際の作業記録を対応付けて解釈したものを，本研究では**プロセスインスタンス**と呼ぶ．また，プロセスインスタンスの部分集合を**サブプロセス**と呼ぶ．

プロセスインスタンスに対しては，作業記録の計測など様々な定量的評価が可能

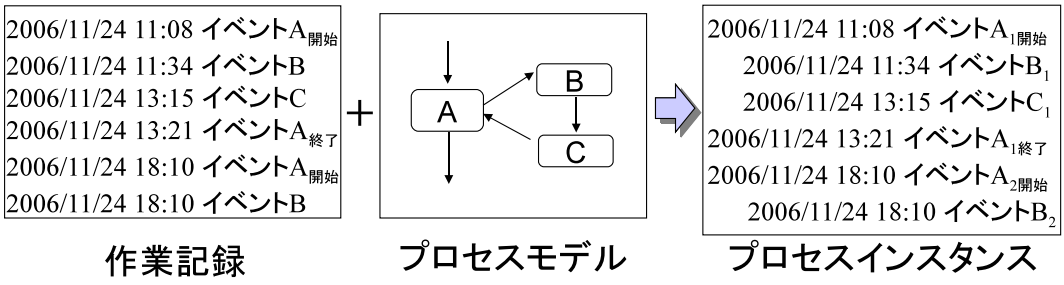

図 6.1　プロセスモデル

である．また，プロセスインスタンスとプロセスモデルを照合し，実際の開発作業がどの程度開発組織内で規定されたプロセスモデル（標準プロセスモデル）に準拠したものであったかを評価することも可能となる．

6.2.3. プロセスメトリクス

一般的にプロセスメトリクスとは，作業効率や生産性といった，抽象的でより粒度の粗い概念である．これに対し，提案手法ではより細粒度のプロセスを取り扱うことで，定量的で粒度の細かい概念を扱うことが可能となる．提案手法により抽出できるメトリクスとしては，下記のようなものが挙げられる．

手順的メトリクス

- 手順漏れ
- 遵守度
- 登録情報漏れ
- 並列度

回数的メトリクス

- 繰り返し回数

時間的メトリクス

- 作業所要時間
- 作業遅延時間

これらのメトリクス自体は，比較的単純なものが数多く含まれるが，いずれも従来の作業報告ベースでのプロセス記録からは計測困難であり，開発作業記録を対象とすることで現実的な計測，評価が可能となる．

6.3. バグ管理システムを利用した保守プロセス

本章では，プロセスメトリクスの抽出，および抽出したメトリクスを利用した分析例として，バグ管理システムを利用したバグ修正プロセスを取り上げる．本節では，前節での分析手法を踏まえ，バグ管理システムを利用したバグ修正プロセスについ

てその作業内容について整理し，いくつかの用語を定義する．

6.3.1. バグ管理システム

GNATS[*1]や Bugzilla[*2]に代表されるバグ管理システムは，ソフトウェア開発プロジェクトにおいて，プロダクトの不具合報告や機能の追加要求を管理するために用いられる．多くのバグ管理システムでは，バグ報告はデータベースに記録され，Web ユーザインタフェースを介して自由に閲覧できるようになっている．バグ管理システムを利用した，典型的なバグ修正のプロセスを以下に示す．

1. バグ票オープン：ユーザがプロダクトの不具合を発見し，バグ票を起票する．
2. 開発者割り当て：バグ管理を受け持つ開発者が，起票されたバグ票に対して，不具合に対応する開発者の割り当てを行う．
3. 不具合修正：割り当てられた開発者が不具合を修正する．
4. 修正確認：修正されたプロダクトが十分に対応されているかを確認する
5. バグ票クローズ：不具合の修正を確認し，対応完了としてマークする．

このようなバグ修正プロセスがどれだけ進行しているかは，バグの状態と呼ばれる．バグの状態をはじめ，割り当てられた開発者の情報は，個々のバグ票に記録される．また，バグを報告する際には，多くの場合，発生した不具合やその再現手順を自然言語で記述する．システムによっては，対象モジュールを選択肢の中から指定する場合や，ファイルの添付が可能な場合もある．また，システム利用者はコメントを随時，自然言語で記述することができる．従って，バグ票を観察することにより，開発者の不具合への対応や，ユーザとのやりとりの状況を把握することができる．

6.3.2. プロセスインスタンス・サブプロセス

本研究では，バグ票 1 件につき，このバグ票に関連するバグ修正プロセスが 1 つ存在すると見なす．また，バグ修正のサブプロセスは，発生したバグを認識・修正する段階である「**修正作業プロセス**」と，修正後，バグが完全に取り除かれたかを確認

[*1] http://www.gnu.org/software/gnats/
[*2] http://www.bugzilla.org/

する段階である「**修正確認プロセス**」の 2 つのプロセスに大別する．

通常，問題なくバグ修正が行われた場合には，プロセスインスタンスには 1 つの修正作業プロセスと，1 つの修正確認プロセスが含まれる．一方，バグ修正が適切に行われなかった場合には，修正確認プロセスの後に再び修正作業プロセスが現れる．すなわち，バグ修正が適切に行われていない場合には，3 つ以上のサブプロセスが含まれる．このように，プロセスインスタンスを時系列に観察した際，修正確認プロセスの後に再び修正作業プロセスが現れた場合を特に「手戻りが発生した」と呼ぶこととする．

6.4. プロセスメトリクスを用いた保守プロセスの分析

6.4.1. 目的

本研究では，ソフトウェア開発プロセスの定量的な評価・分析・改善手法の確立を目的とする．近年のように，開発サイクルが短期になっている場合，その評価・分析を可能な限り迅速に行い，次回以降のプロジェクトの改善に活かすことが求められている．そのため，本研究では，特に自動収集された開発データを定量的に分析することで，データの収集・分析コストをかけずに開発プロセスの品質評価・改善を行う手法の確立を目指す．

その一環として，本研究では，ソフトウェアのバグ修正プロセスをケーススタディとして取り上げる．このケーススタディの目的は，提案手法を用いて実際の開発記録からプロセスメトリクスが抽出できることを確認することと，プロセスメトリクスを用いることで開発プロセスの分析が可能になることを示すことである．プロセスメトリクスの算出にあたっては，多くの開発プロジェクトで採用されているバグ管理システムに登録されているデータを用いる．以降，6.4.2 節でバグ管理システムから抽出できる保守プロセスを特徴づけるプロセスメトリクスについて述べ，6.4.3 節でプロセスメトリクスを用いることで確認できると思われる開発プロセスの特徴を仮説としてまとめる．

6.4.2. プロセスメトリクス

6.3.1 節で述べたように，多くのバグ管理システムで管理されているバグは，個々のバグがどのような状態にあるかを記録している．これらの状態を観察することで，バグ修正時にどのような作業が行われたかを確認することができる．

本研究では，この状態の数や，修正にかかった時間に着目し，以下のプロセスメトリクスを提案する．

作業時間　バグ票が起票されてから，修正が完了するまでの時間
状態遷移回数　バグの状態が変更された回数
作業手戻り回数　確認作業を表す状態から修正作業を表す状態に，バグの状態が変更された回数
遵守度　標準プロセスモデル（図 6.2）に沿って作業をしているかどうか
作業者数　バグにかかわった人員の数

6.4.3. 仮説

ケーススタディでは，バグ修正プロセスにおいて修正作業が適正に行われたかどうかをプロセスメトリクスを用いることで判別できるかどうかを検証する．作業が適正に進行したかどうかの判断基準の一つとして，本研究では，バグ修正作業において作業の手戻りが発生したかを取り上げる．バグ修正は 6.3.1 節で述べたように，通常であれば，修正が行われた後，修正が適正に行われたかが確認され，バグはクローズされる．しかし，修正プロセスにおいて修正作業が適正に行われなかった場合には，確認プロセスの後，再び修正プロセスに移行する，すなわち手戻りが発生すると考えられる．

本研究では，まず提案するプロセスメトリクスを用いて，このような手戻りが発生しているプロセスを弁別できるかを確認するため，以下の仮説を立てる．

仮説 1　バグ修正プロセスの中には，手戻りが発生しているものが含まれる．

手戻りが発生するのは，該当するプロセスで修正されているプロダクトに原因が

あると考えられる．すなわち，プロダクトの何らかの要因が，修正プロセスの正常な実施を阻害していると考えられる．このような要因の一つとしては，修正対象のファイル数の多寡が考えられる．これは，修正対象のファイル数が多いと，修正範囲が大きくなり，結果として十分にバグを取り除くことができなくなると思われるためである．これより，以下の仮説が導かれる．

仮説 2　手戻りが発生しているプロセスで扱われているファイルの数は，手戻りが発生していないプロセスに比べて多い．

また，手戻りが発生するようなプロセスで扱われているファイルは，該当バグが修正された後，違うバグを引き起こす可能性があるとも考えられる．そこで，以下の仮説を立てる．

仮説 3　手戻りが発生しているプロセスで扱っているファイルは，バグが多い（複数のバグ修正プロセスに出現する）ファイルである

次節では，上記の仮説を確かめるため実施したケーススタディについて報告する．

6.4.4. 対象データ

本研究では，6.4.3 節で示した仮説を検証するため，オープンソースソフトウェアの統合開発環境である Eclipse プロジェクトのデータを対象とした．Eclipse プロジェクトでは，バグ管理に Bugzilla が利用されている．本ケーススタディでは，この Bugzilla に登録されたバグ票のうち，Zimmermann らの研究 [79] の成果として公開されているデータに記載されている，バグ票に対応する修正箇所が明らかになっているものを利用した[*3]．これらのバグ票は，Eclipse 2.0，2.1，3.0 の各リリース日前後半年の間に修正作業が行われたものである．表 6.1 に，各バージョンのリリース日と，観察対象となったバグ票の件数を示す．

またこのデータには，ファイルおよびパッケージごとに，コード行数や複雑度といったプロダクトメトリクスや，リリース前後のバグ数が掲載されている．リリース

[*3] `http://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/` で公開されているデータを 2009 年 6 月 15 日に取得し，利用した．

表 6.1　観察対象としたバグ票

バージョン	2.0	2.1	3.0
件数	3401	2480	4136
リリース日	2002/06/27	2003/03/27	2004/06/25

前後のバグ数とプロセスメトリクスの関係を分析するにあたっては，これらのデータも利用した．

Zimmermann らが公開しているデータには，バグ票の状態の変更履歴や，その変更が行われた時刻，変更した人物に関する情報は記録されていない．そこで，これらのデータに関しては，Eclipse プロジェクトの Bugzilla[*4]にアクセスし，分析対象としたバグ票の詳細な変更履歴を取得した．

6.4.5. Bugzilla におけるバグ票の状態遷移

Bugzilla のバグ票には，様々な情報を記入するためのフィールド（欄）が用意されている．本研究では，特に “Status” 欄に着目して分析を行う．“Status” 欄には，各バグ票の状態が逐次記録される．この欄の履歴を分析することにより，各バグ票においてどのような状態変化が起きたかを収集することができる．

Bugzilla で管理されているバグ票は，図 6.2 のような状態をとる．図 6.2 の状態遷移は，6.3.1 節で示した，一般的なバグ修正プロセスに沿ったものとなっている．バグが発見されると，報告者によりバグが報告され（「未承認」），その報告が妥当なものであれば状態が「報告」となる．報告されたバグは，修正作業担当者が割り当てられ（「割り当て」），修正が行われた後，「修正」状態となる．修正されたモジュールは，品質保証グループなどによって修正が正しく行われたか確認され（「確認」），正しく修正されていれば修正作業は完了となる（「終了」）．その後，バグが正しく修正されていなかった場合には，「再発」状態に移行し，再び修正作業が行われる．

このように，バグは図 6.2 に示す 7 つの状態をとる．さらに，この 7 つの状態は大きく分けて，修正作業中を示す状態（「未承認」「報告」「割当」「再発」）と修正確認

[*4] https://bugs.eclipse.org/bugs/

中/修正完了を示す状態（「修正」「確認」「終了」）の 2 つに分けられる．

6.4.6.　分析手順

6.4.3 節で示した仮説を検証するため，6.4.2 節で示したプロセスメトリクスをバグ票から抽出する．次に，関連するファイルおよびこれらのファイルに含まれるリリース前後のバグ数を抽出する．最後に，これらの抽出した定量的な値を分析する．以下に手順の詳細を示す．

プロセスメトリクスの抽出

6.4.4 節で示したデータセットに対して，以下の手順でプロセスメトリクスを抽出し，分析を行った．

1. 個々のバグ票に記録された状態変更の記録を時系列に観察し，どのような状態遷移が行われたか，プロセスインスタンスを抽出した．
2. (1) で抽出したプロセスインスタンスから，サブプロセス（修正作業プロセスと修正確認プロセス）を弁別した．
3. サブプロセスのうち，修正確認プロセスから修正作業プロセスに移行した回数を，「手戻り回数」として数え上げた．

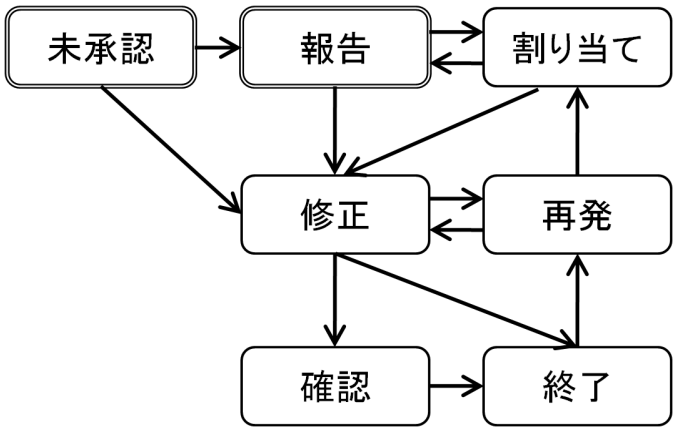

図 6.2　バグ票の状態

関連ファイル・バグ数の抽出

先の手順で抽出された各プロセスインスタンスについて，以下の手順で，そのプロセスで扱われているファイル，およびそのファイルのバグ数を抽出する．

1. プロセスインスタンスごとに，6.4.4 節で示したデータセットを利用して，対応するバグが修正されたと記録されているファイルを抽出する．
2. 抽出されたファイルごとに，6.4.4 節で示したデータセットからリリース前後に対応されたバグの数を算出する．
3. 個々のプロセスインスタンスで扱われていたファイルの，リリース前後に発見されたバグ数の平均値を算出する．

上記手順について，具体例を用いて説明する．今，プロセスインスタンス 1 で対応されたファイルが A，B の 2 つであり，それぞれ，リリース前に 6 個，4 個，リリース後に 3 個，2 個のバグが修正されたとする．このとき，プロセスインスタンス 1 で対応されたファイル A，B を「プロセスインスタンス 1 に関連するファイル」と呼ぶ．また，この関連するファイルのバグ数平均値は，リリース前が $(6+3)/2 = 4.5$ 個，リリース後が $(4+2)/2 = 3$ 個となる．

仮説の検証

上記の手順で抽出された，プロセスメトリクスとプロダクトメトリクスの値を，統計的に分析することで，仮説の検証を行う．

6.4.7. 結果

プロセスメトリクスとして，図 6.3 に各バグ票の状態遷移回数の分布を，表 6.2 に作業手戻り回数の分布を示す．また，表 6.3 にバグ修正プロセスにおける作業の遵守度を示す．

表 6.2 より．作業の手戻りはほとんどのバグ票では発生していない一方で，最大で 5 回にわたり手戻りが発生しているケースが確認された．

次に，各バグ票について，個々のバグ票に関連する（すなわち，報告されたバグの影響を受けていた）ファイルの数を求めた．図 6.4 にバージョンごとの関連するファ

表 6.2　バグ収集プロセスの手戻り回数

手戻り回数	0	1	2	3	4	5	合計
Ver. 2.0	2809	487	80	16	7	2	3401
Ver. 2.1	2156	264	47	11	2	0	2480
Ver. 3.0	3501	522	90	21	1	1	4136

イル数の分布を示す．多くのバグ票では，1 つのバグが影響するファイルは 1 つのファイルだが，まれに数十から数百のファイルに影響を及ぼすバグがあることがわかる．

また，手戻りが発生したプロセスと発生しなかったプロセスの 2 群に分け，1 つの

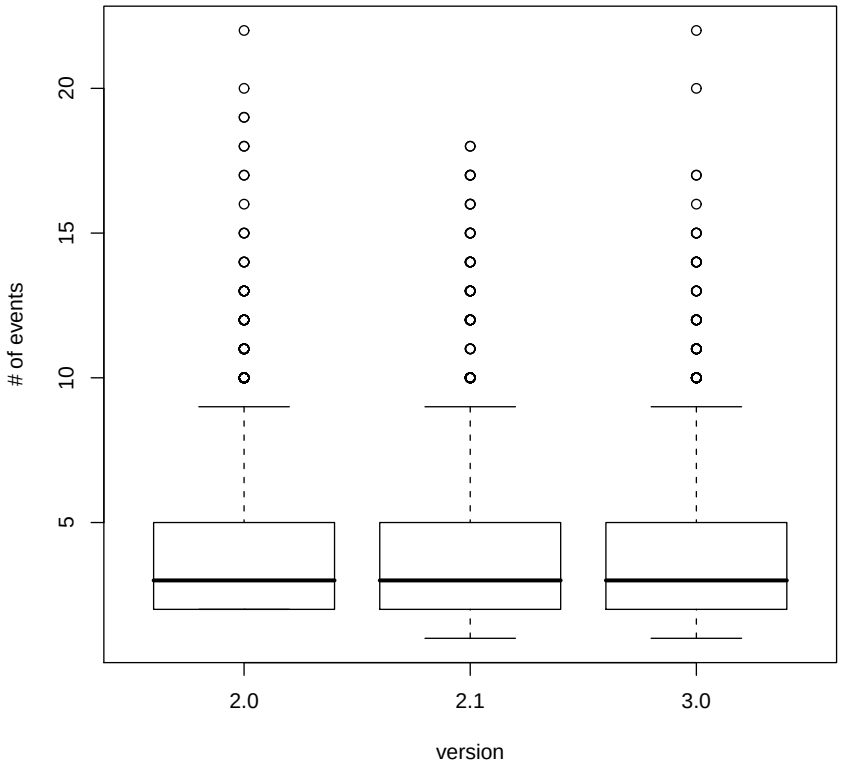

図 6.3　バグ票の状態遷移回数

表 6.3　バグ収集プロセスの作業遵守度

バージョン	遵守していたバグ票	遵守していなかったバグ票	合計
Ver. 2.0	3006	395	3401
Ver. 2.1	2262	218	2480
Ver. 3.0	3732	404	4136

バグが影響するファイル数に関して，2 群の間で差があるかを Mann Whitney の U 検定を用いて検証した．表 6.4 にファイル数の平均値と検定の結果を示す．表 6.4 より，2 群の間には統計的に有意な差があり，手戻りが発生したプロセスで扱われているバグの方が，わずかに影響するファイル数が多いことがわかる．

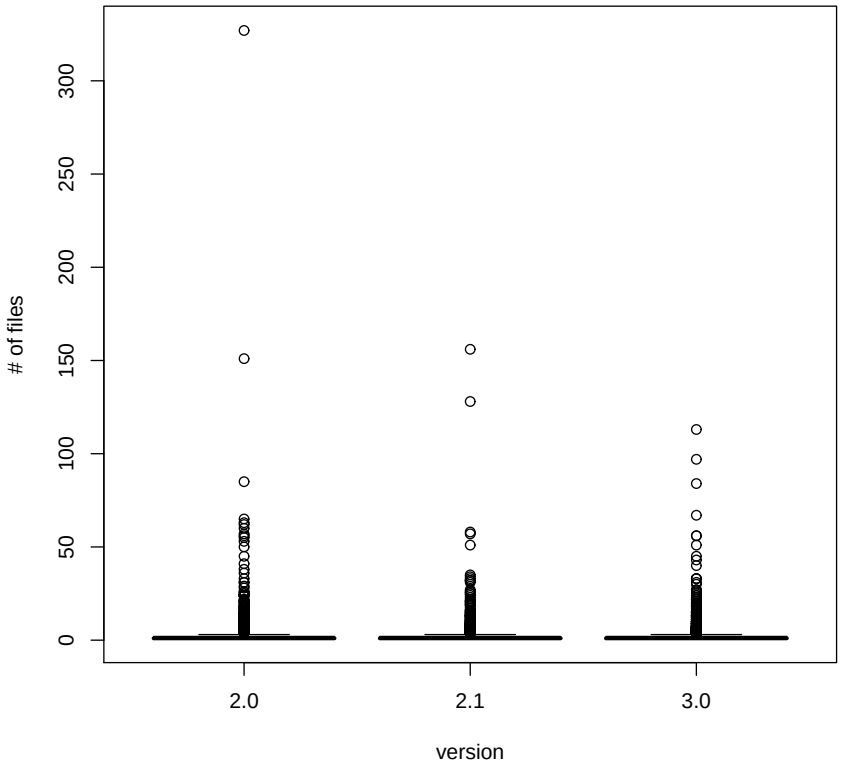

図 6.4　バグの影響を受けたファイルの数

表 6.4　ファイル数の差

バージョン	平均値		p 値
	手戻りあり	手戻りなし	
Ver. 2.0	3.2	2.6	0.05
Ver. 2.1	3.5	2.3	0.00
Ver. 3.0	3.4	2.3	0.00

表 6.5　リリース前のバグ数の差

バージョン	平均値		p 値
	手戻りあり	手戻りなし	
Ver. 2.0	6.0	6.2	0.71
Ver. 2.1	5.3	5.3	0.65
Ver. 3.0	6.8	7.0	0.42

最後に，プロセスインスタンスで扱われているファイルが，どの程度バグを含んでいたのかを確かめた．各ファイルに含まれていた，各バージョンのリリース前およびリリース後に修正されたバグの数は，Zimmermann らの分析により明らかになっている．そこで，各プロセスインスタンスで扱われていたファイルについて，1 ファイルあたりのリリース前およびリリース後のバグの数を算出した．そして，バグが影響したファイル数に関する分析と同様に，手戻りの発生の有無によって，リリース前後のバグ数に差があるかを Mann Whitney の U 検定を用いて検証した．リリース前後のバグ数の平均値と検定の結果を表 6.5 および表 6.6 に示す． 表 6.5 より，手戻りが発生したプロセスよりも，手戻りが発生しなかったプロセスで扱っていたファイルの方が，わずかにリリース前に修正されたバグを多く含んでいることがわかる．しかし，統計的に有意な差ではなく，その平均値もほとんど差がない．一方，表 6.6 より，バージョン 2.0 前後およびバージョン 2.1 リリース前後のバグ修正プロセスでは，手戻りが発生したプロセスで扱われたファイルが，リリース後に発見されたバグをわずかに多く含んでいることがわかる．また，これらの差は統計的に有意な値が

表 6.6　リリース後のバグ数の差

バージョン	平均値		p 値
	手戻りあり	手戻りなし	
Ver. 2.0	2.8	2.3	0.08
Ver. 2.1	1.3	1.1	0.00
Ver. 3.0	2.0	2.1	0.85

示されている．

6.5. 考察

6.5.1. 分析結果の検証

表 6.2 より，バグ修正プロセスにおいて，8 割程度のプロセスは手戻りが発生せずバグ修正が行われていることがわかる．この結果は，仮説 1 が成り立つことを支持している．また，手戻りが発生していたとしても，複数回手戻りが発生していることはまれである．このことから，手戻りが発生した場合は，バグ確認プロセスが有効に機能し，バグが完全に修正されていないにも関わらず，修正と判断するような不完全な修正を防止していると考えられる．

ただし，複数回手戻りが発生しているようなプロセスでは，バグ修正において何かしらの問題が発生していることが予想される．実際に，今回使用したデータセットの中で最も手戻りが発生していたプロセス（バグ#54989 に関するプロセス）に関して Bugzilla のコメント欄を分析した．その結果，コメント欄には 50 を超える開発者間のやりとりが記録されていた．この中では，あるモジュールの挙動に関するバグ報告に対して，旧バージョンと現行バージョンでの挙動の違いに対する認識の違いや再現手順の不明確さから，修正作業が混乱していることが読み取れる．今回提案するプロセスメトリクスの一つである手戻り回数は，このようなバグ修正プロセスにおける，混乱の度合いを測る指標として使える可能性がある．

表 6.4 の結果から，手戻りが発生する要因の一つとしては，プロセスで扱っているバグの修正対象範囲（ファイル数）の影響がある可能性がある．修正対象が大きくな

ると，その修正作業が複雑となり，結果として十分に修正が完了しないまま修正確認プロセスへと移行し，その際に修正漏れが発見され手戻りが発生していると考えられる．ただし，手戻り発生の有無によるファイル数の差は，統計的に有意ではあるがわずかである．この結果は，ファイル数の差はわずかではあるが，仮説 2 が成り立つことを支持していると考えられる．修正作業の複雑さという観点からは，単純なファイル数の議論だけではなく，複雑度メトリクスなどを併用して，より詳細に修正対象のプロダクトの特性が，プロセスに与える影響を分析する必要がある．

表 6.5 および表 6.6 の結果から，バージョン 2.1 の場合のみ，手戻りが発生したプロセスで扱われていたファイルは，手戻りが発生していないプロセスで扱われたファイルと比べて，リリース後に発見されるバグの数がわずかではあるが，有意に多いことがわかる．

このことから，バージョン 2.1 のときに限り，手戻りが発生しているようなプロセスで扱われていたファイル（プロダクト）は，そうでないファイルと比べて，バグが多く残存していると考えられる．また，仮説 3 はバージョンが 2.1 のときのみ成立しているが，バグの数の差はわずかであると言える．先にも述べたように，手戻りが発生している場合には，そのプロセスが混乱している可能性もあることから，混乱したプロセスで修正されたプロダクトは，修正後も別の欠陥を含むような，品質の悪いものである，と考えることもできる．

今回分析した結果得られた知見を利用することで，開発プロジェクト実施中にプロダクトの品質を予測できる可能性がある．具体的には，ファイル数と手戻り回数の関係や，手戻り回数とリリース後のバグ数の関係をもとに，リリース後の残存バグ数が多いと思われるプロダクトを特定し，そのプロダクトに大して集中的にテストを行う，といった利用が考えられる．

6.5.2. 手法の適用可能性

提案する分析手法は，開発組織内で開発プロセスに関する作業記録が残されていることが前提となっている．分析にあたっては，対象となるプロセスでの作業記録を選別し，この作業記録から分析に必要なプロセスメトリクスを抽出する必要がある．プロセスメトリクスの抽出にあたっては，必ずしも収集作業が自動化されている必要はない．しかし，今回のケーススタディで示したように，分析にあたっては

大量のデータが必要となる可能性がある．そのために，分析，評価のためのコスト削減と言う観点からは，作業記録が自動収集されていることが望ましい．開発作業記録の自動収集ツールを利用することで，データ収集コストの削減を達成した上で，簡便に定量的なプロセス評価を行うことができる．

開発作業記録に関しては，ソースコードの改版情報や本研究で扱ったバグの修正記録など，下流工程を中心に多くの組織で自動的な収集が行われている．このような組織に対しては，本手法を適用することができる．また，自動収集を行っていない組織に関しては，提案手法を用いて開発作業記録からのプロセスメトリクス抽出が可能であることを確認した上で，対象となる作業記録が自動収集することが可能かを検討して行くことができると考えられる．

6.5.3. 従来手法との比較

これまでの開発プロセスの分析・評価手法は，CMMI のように組織のプロセス実施能力を定性的に評価するものか，成果物の管理情報として記録されたデータからプロセスに関する情報を取り出し，作業効率や生産性についての定量的な評価を行うものである．CMMI のような定性的なプロセス評価では，組織内で実施された開発作業に関する記録をもとに，人手によって開発プロセスの評価を行う．これに対して，提案手法では，開発作業から自動的に抽出することが可能なメトリクスを用いて，属人性を排除した定量的なプロセス評価を行うことができる．また，提案手法は開発作業記録を用いて，自動的にプロセスに関する情報取り出すことで評価を行う．ここで抽出される情報は，従来と比べて，作業単位のより細粒度なプロセスに関する情報を抽出することができる．

これまでに提案されている開発プロセスの分析・評価手法と比較して，提案手法の利点はより簡便に定量的な評価を行うことができる点にある．従来の手法は，分析にあたって必要なデータの収集が煩雑であった．また，収集したデータは通常人手によって分析するため，分析者に対する教育コストがかかることも問題であった．本提案手法では，開発記録を自動収集するツールを利用し，データ収集コストの削減を図った．このような自動収集ツールの利用は以前から行われていたが，収集されたデータをどのように分析することで開発プロセスの評価が可能となるかは明らかになっていなかった．本研究の貢献は，このような開発作業記録を用いた開発プロセス

の分析，評価の方法を体系立てて整理し，低コストな分析手法を確立した点にある．

6.6. 関連研究

開発プロセスの分析・評価は，ソフトウェア開発において重要となっており，これまでに多くのパラダイムが提案されている．例えば，Basili らの提唱する GQM (Goal - Question - Metrics)[9] や McGarry らの提案する PSM (Practical Software Measurement)[52] 等の方法論，カーネギーメロン大学ソフトウェア工学研究所が開発した CMMI (Capability Maturity Model Integration)[13] や Humphry が提唱した PSP (Personal Software Process)[30] などのフレームワークが存在する．このようなプロセスを評価する取り組みの多くは，組織のプロセス実施能力を定性的に評価するものか，成果物の管理情報として記録されたデータからプロセスに関する情報を取り出し，作業効率や生産性についての定量的な評価を行うものである．これに対し，本研究は，開発中に自動収集されるデータを用いて，低コストで開発プロセスの品質を定量的な評価を目指している．

本研究では，開発記録をもとにソフトウェアプロセスを時系列に分析した．我々は，開発記録を用いて，開発プロセスを細かい粒度で分析する手法を「マイクロプロセス分析手法」として提案している [57]．また，同様のアプローチを用いて，あらかじめ規定された作業手順を遵守しているかによって，バグの修正作業に影響が出るか分析している [19]．さらに，我々はバグの修正作業におけるユーザと開発者の間の認識のずれを検出する手法についても提案している [44]．我々のアプローチと類似した研究としては，花川はソースコードの複雑さに影響を与えるマイクロプロセスを抽出する手法 [27] を提案されている．また伊原らは，バグ管理システムにおける，バグの初期対応時間と滞留時間の関係について分析をしている [32]．

ソースコードの追加・削除行数や作業の滞留時間といった，開発作業に関して自動的に収集可能なメトリクスも多く提案されている [24, 54, 66]．今回の分析でも利用した，Zimmermann らの研究では，Eclipse の各プロダクトに含まれる，リリース前後のバグ数とプロダクトメトリクスとの相関分析を行っている．また，Moser らは，複雑度メトリクスなどのプロダクトメトリクスと，版管理システムの履歴を分析することで抽出できるメトリクス（コードの追加・削除行数）などを併用してバグ予測

モデルを構築することで，プロダクトメトリクスのみでモデルを構築するよりもより良い精度が出ると報告している [58].

バグ管理システムは，オープンソースソフトウェア開発においても利用されることが多い．そのため，オープンソースソフトウェアの開発プロジェクトを分析する際には，バグ管理システムや版管理システム，メーリングリストなどのソフトウェアリポジトリを対象とする場合が多い．Mockus らは，これまでオープンソースソフトウェア開発について指摘されていた事柄の正しさについて，CVS とバグ管理システムのデータを調査した [54].

6.7. おわりに

本章では，自動的に収集された開発作業記録を利用したソフトウェア開発プロセスの定量的な分析・評価手法を提案した．提案手法では，開発作業記録から開発プロセスを抽出し，標準プロセスモデルと比較することで，開発プロセスの特徴を表すメトリクス（プロセスメトリクス）を算出する．そして，このプロセスメトリクスを用いて開発プロセスの品質を分析する．

提案手法を用いて，実際の開発作業記録からプロセスメトリクスが抽出できること，プロセスメトリクスを用いて開発プロセスの分析が可能であるか，の 2 点を確認するために，Eclipse プロジェクトのバグ修正プロセスを対象に分析を行った．分析にあたっては，バグ管理システムに記録されているバグの状態履歴を利用し，プロセスの手戻りに着目して分析を行った．分析の結果，ファイル数やリリース後に発見されたバグ数についてグループ間で比較した結果，手戻りの発生したプロセスが，わずかではあるがより多くのファイルが扱われていることがわかった．また，限定的ではあるが，リリース後に発見されたバグも，手戻りの発生したプロセスで扱われたファイルの方が，より多く含むことがわかった．

今回実施した分析の結果は，提案手法が従来の手法と比較して，より簡便に開発プロセスの分析を定量的に行うことができることを示唆している．今後の展望としては，複雑度などのプロダクトメトリクスと今回提案したプロセスメトリクスの相関関係を分析することにより，プロセスの品質とプロダクトの品質が互いにどのような影響を与えているかについて分析をすることが挙げられる．また，関連するソースコー

ドなどの開発ドキュメントを用いて，提案するプロセスメトリクスで，開発プロセスのどのような特徴を示すことができるのかについて検討を行うことも挙げられる．

第 7 章

リポジトリデータを用いたデザインパターン適用履歴分析手法

7.1. はじめに

デザインパターン [20] はソフトウェア開発における設計のベストプラクティスをまとめたものである．デザインパターンを用いることにより，ソフトウェアの設計が洗練され，また要求分析時とのギャップが埋まり，最終的にソフトウェアの品質を向上させることができると言われている．デザインパターンを利用する場面は，初めからデザインパターンを念頭に置いて設計・実装を行う場合もある．しかし多くの場合は，実装時に何らかの問題に直面し，その問題の解決策としてデザインパターンを利用することが多い．また，デザインパターンを利用する際には，一回デザインパターンを適用して終わるだけとは限らない．工程が進むに従って，さらに洗練された設計へと進化させるために，異なるデザインパターンに変更する場合や，適用していたデザインパターンを取り除く場合がある．それゆえ，デザインパターンがソフトウェアの品質に与える影響を分析するには，ソフトウェアの時系列分析を行うことが有益であると考える．

そこで，本研究では，デザインパターンを利用することで開発プロダクトに与える影響を分析するための手法として，版管理システムを用いたデザインパターンの適用履歴分析を提案する．この手法を用いることで，ソフトウェアの開発時にデザインパターンがどのように利用され，どのような影響を与えているかを分析すること

が可能となる．

以下，2 節ではデザインパターンに関する研究について述べ，3 節ではデザインパターンに関して形式的な定義を与える．4 節で提案する版管理システムを用いたデザインパターンの分析手法について述べ，5 節で提案手法を適用して行った実験について報告する．6 節で関連研究について述べ，7 節でまとめを述べる．

7.2. 背景

デザインパターンに関する研究は多数行われている．ここではその中でも，デザインパターンを適用する際に見られる，デザインパターンの関連に関する研究について述べる．また，工程が進むにつれ，どのようにデザインパターンが進化するかについて述べた研究についても述べる．

表 7.1 GoF のデザインパターン間の関連（抜粋）

パターン名	代用できるパターン	同時に使われるパターン
Abstract Factory	Builder，Prototype	Singleton
Chain of Responsibility	State	Composite，Decorator
Composite	Builder，Visitor，Decorator，Iterator	Flyweight

デザインパターンを適用する際，デザインパターンによっては，他のデザインパターンで代用できるものや，同時に用いるべきデザインパターンが示されている場合がある [78]．表 7.1 に GoF のデザインパターンでの，デザインパターン間の関連を示す．ここでは，あるデザインパターンを適用する際に，同時に利用すべきデザインパターンや類似のデザインパターンが記されている．例えば，Abstract Factory パターンは，Builder パターンと類似している．また，このパターンの構成要素の一つである ConcreteFactory オブジェクトは，Singleton パターンを適用して作られたオブジェクトである．このようにデザインパターンは，それぞれが個別に適用されるのではなく，状況に応じて複数のパターンを組み合わせて適用する場合がある．

デザインパターンは，リファクタリング [18] を行う過程で適用されることがある．しかし，デザインパターンは一回適用された後，修正されないわけではない．場合によっては，より洗練された設計とするために，違うデザインパターンを適用する場合や，デザインパターンを取り除く場合もある．

表 7.2　デザインパターンごとのリファクタリング

パターン名	パターンを取り入れる	パターンに近づく	パターンから離れる
Composite	Composite による単数・複数別の処理の置き換え Composte の抽出	−	Builder による Composite の隠蔽
Iterator	−	−	Visitor による累積処理の書き換え
Singleton	Singleton による状態変化のための条件判断の置き換え	−	Singleton のインライン化
Adapter	Adapter によるインタフェースの統合 Adapter の抽出	Adapter によるインタフェースの統合	−

Kerievsky は，デザインパターンを取り入れたリファクタリングについてまとめている [48]．この中で，Kerievsky は，単にデザインパターンを取り入れるだけでなく，別のパターンに変更する場合や，パターンを取り除く場合などのケースをまとめている．表 7.2 にパターンごとのリファクタリングを示す．例えば，Composite パターンでは，単数・複数別の処理を Composite パターンによって置き換える（Composite パターンを取り入れる）以外に，Builder パターンを Composite パターンと置き換え，オブジェクトの構築処理を単純化するといったリファクタリング方法が示されている．

これらの研究では，実装時にいくつかのクラス間の関係が修正されることが指摘されているが，それがどのように行われているか，実証している研究は我々の知る限り存在しない．このような事実を実証するには，デザインパターンがどのように適用されているかを時系列に分析する必要がある．そこで，我々はデザインパターンの時系列分析手法を提案する．

7.3. 定義

7.3.1. デザインパターン

ある時刻 t におけるソースコードファイルの集合を F_t とする．またデザインパターンが掲載されているデザインパターンカタログを cat とする．本研究ではデザインパターン dp を次式で定義する．

$$\begin{aligned} dp &= (n, R), \\ &\text{where } n \in \{\text{design pattern name in } cat\}, \\ &\qquad R = \{\text{participants of design pattern } n \text{ in } cat\}. \end{aligned}$$

デザインパターンは，その名前 n とデザインパターンの構成要素の集合 R から成る．また，cat に掲載されているデザインパターンの集合を PC で表す．表 7.3 に GoF のデザインパターンでの例を示す．

表 7.3 GoF のデザインパターンにおけるパターンとパターンを構成する役割（抜粋）

パターン名	役割
Factory Method	Product，ConcreteProduct，Creator，ConcreteCreator
Singleton	Singleton
Observer	Subject，ConcreteSubject，Observer，ConcreteObserver

また，dp から名前を取り出す演算を，f_{name}，役割を取り出す演算を f_{role} で定義する．

デザインパターンインスタンス dpi は，デザインパターンを実際のプロダクトに適用したもので，次式で定義する．

$$\begin{aligned} dpi &= (dp, DPP, h), \\ &\text{where } dp \in PC, \\ &\qquad DPP = \{\text{design pattern pieces}\}, \\ &\qquad h : f_{role}(dp) \to DPP. \end{aligned}$$

デザインパターンインスタンスは，カタログ中のデザインパターン dp，デザインパターンの構成要素に相当するクラス群 DPP，dp と DPP 間の写像関係 h の組で表

す．ここで，デザインパターンの構成要素に相当するクラスを特に，デザインパターンインスタンスと呼ぶ．さらに，dpi から名前 n を取り出す演算を，g_{name} で定義する．

上記のデザインパターンインスタンス dpi で利用されている，デザインパターンピース dpp の定義を以下に示す．

$$\begin{aligned} dpp \quad &= (r, c), \\ &\text{where } r \in \{x | x = f_{role}(dp), dp \in PC, \\ &\qquad\qquad f_{name}(dp) = g_{name}(dpi)\}, \\ &\qquad c \in \{\text{classes of } F_t\}. \end{aligned}$$

デザインパターンピースは，個々のデザインパターンインスタンスにおいて，それぞれのデザインパターンの役割名 r とその役割に対応するクラスから成る．

本研究では，開発プロダクト中のデザインパターンをとらえる単位として，デザインパターンインスタンスを採用する．すなわち，開発プロダクトの各時点において，どのようなデザインパターンインスタンスが開発プロダクト中に含まれているかを測定する．今，時刻 t において F_t に適用されたデザインパターンインスタンスの集合を P_t とする．本研究では，各時点における P_t の変遷を観察することで，開発プロダクトに適用されたデザインパターンがどのように変化していったかを考察する．以下，分析にあたって，デザインパターンインスタンスの集合 P と P' との集合演算を以下のように定義する．

$P \cap P'$： P と P' のどちらにも属するデザインパターンインスタンスを求める．

$P' - P$： P' には存在するが，P には存在しないデザインパターンインスタンスを求める．

7.3.2. デザインパターン適用履歴

デザインパターンが開発プロダクトに適用された履歴を分析するためには，時刻 t と $t-1$ におけるデザインパターンインスタンスの集合 P_t と P_{t-1} を比較する必要がある．ここで，以下の表記を導入する．

PA_t： 時刻 t で追加されたデザインパターンインスタンスの集合．$P_t - P_{t-1}$ でもとまる．

PD_t: 時刻 t で削除されたデザインパターンインスタンスの集合．$P_{t-1} - P_t$ でもとまる．

PR_t: 時刻 t，$t-1$ のどちらでも存在していたデザインパターンインスタンスの集合．$P_{t-1} \cap P_t$ で求まる．

また，自明であるが，$P_t = \{PR_t \cup PA_t\}$, $P_{t-1} = \{PR_t \cup PD_t\}$ である．

時刻 t における，P_t と P_{t-1} とのデザインパターン履歴関係 H_t を次式で定義する．

$$H_t = (PA_t, PD_t, PR_t)$$

最後に，時刻 t から $t + \Delta$ までのデザインパターン履歴関係の集合を次式で定義する．

$$History_{[t,t+\Delta]} = (H_t, H_{t+1}, \cdots, H_{t+\Delta})$$

7.4. デザインパターン進化分析手法

本節では，提案するデザインパターンの適用履歴を調査する手法について述べる．始めに提案手法で前提としているシステムについて概説し，その後に提案手法の詳細について述べる．

7.4.1. 検出に用いるシステム

デザインパターン検出手法

デザインパターンの履歴を分析するにあたって，開発プロダクトに使われているデザインパターンを知るには，対象プロダクトの設計ドキュメントを読むことが考えられる．しかし，設計ドキュメントが十分に整備されていないケースがしばしば見られ，そもそもドキュメントが存在しない場合もある．また，設計ドキュメントが残されている場合にも，具体的にどのようなデザインパターンが開発プロダクトに適用されたか明記されているケースは少ない．そのため，デザインパターンの履歴分析にあたっては，開発プロダクトに適用されているデザインパターンを検出する手法が必要となる．

デザインパターン検出手法は，既に数多く提案されている．提案されてる検出手法は，主にオブジェクト指向型プログラミング言語からのデザインパターン検出を目的

としたものがほとんどであり，Java を対象としたもの [28] や C++ を対象としたもの [6] などがある．これら多くの手法では，事前情報として GoF などの既知のデザインパターンを記憶しておき，その事前情報に基づきソースコード中からデザインパターンを分析する．検出にあたっては，静的解析に基づくプログラムの構造を利用したものだけでなく，メソッド呼び出し関係などの動的解析に基づくものもある．

本研究では，Tsantalis らが提案する検出手法 [74] を採用する．この手法は，グラフ理論と類似度計算を用いたデザインパターン検出手法であり，Java で記述されたプログラムを静的解析することで，デザインパターンを検出する．この手法では，クラスをグラフの頂点，クラス間の関連（継承関係など）をグラフの有向辺としてグラフを構築する．そして，検出対象のソフトウェアから抽出されるグラフとデザインパターンのグラフとを比較し，類似部分を検出することで，ソフトウェアに適用されているデザインパターンを検出する．この手法を本研究で採用した理由は，Tsantalis らの提案手法が比較的高速なアルゴリズムであり，パターンカタログに記載されているデザインパターンだけでなく，カタログに記載されたものから多少修正して適用されたデザインパターンも検出することが可能であるためである．また，彼らの手法を実装したツール [2] が公開されており，容易に利用可能なためである．なお，このツールでは，Gamma らが提唱する 23 のデザインパターン [20] のうち，10 種類のデザインパターンが検出可能となっている．ただし，このツールは対象ソフトウェアのソースコードを必要とする．また，検出時に Java のリフレクション機構を利用するため，ソースコードがエラーなくコンパイルでき，実行できる必要がある．

ソースコード管理システム

ソースコード管理システムとは CVS や Subversion のようにプロダクトの保管・管理に用いられるシステムを指す．これらのシステムでは，管理下のプロダクトを任意の時点の状態に復元して取得する機能が提供されている．

時系列にデザインパターンが進化する様子を観察するためには，過去のソースコードにおけるデザインパターンの適用状況を解析することが必要である．このような情報を得るためには，版管理システムを用いて過去のソースコードを取得し，そのソースコードに対してデザインパターン分析手法を適用すれば良い．本研究では，Subverison を採用する．Subversion は多くの開発プロジェクトで採用されている

ソースコード管理システムであり，アトミックなコミットをサポートしている．アトミックなコミットがサポートされていることにより，同時に複数の修正がなされたとき，それらの修正は 1 つの修正作業として記録される．デザインパターンが適用される場合，複数のファイルが同時に修正が行われることがある．ある時点において同時にどのようにデザインパターンが修正されたかを分析するためには，このようなアトミックなコミットをサポートしているシステムに記録されているデータを用いる必要がある．

7.4.2. 分析手順

以下にデザインパターン進化過程分析手法の手順を示す．デザインパターン履歴分析を適用する期間の始まりを時刻 0，終わりを時刻 T で表す．F_t を時刻 t におけるファイルの集合，P_t を時刻 t におけるデザインパターンインスタンスの集合とする．

1. $History_{[0,T]} = \emptyset$
2. $t = 0$ の F_0 を版管理システムから取得し，デザインパターン検出手法を用いて P_0 を求める．
3. for $t = 1, 2, \cdots, T$
   (a) 版管理システムから F_t を取得し，デザインパターン検出手法を用いて P_t を求める
   (b) P_t と P_{t-1} を比較し，H_t を求める．
   (c) $History_{0,T}$ に H_t を追加する．

このように，本手法では分析を行う期間 $[0, T]$ に対して，デザインパターン履歴関係 $H_1, \cdots, H_t$ を順次抽出する．分析を行う際には，上記の手順で抽出したデザインパターン履歴関係 $History_{0,T}$ を用いて分析を行う．

7.4.3. デザインパターン適用履歴に着目したメトリクス

前節で示した分析手法を用いることで，ソフトウェアに修正が加えられた際に，デザインパターンの適用状況がどのように変化したかを分析することが可能となる．

表 7.4　提案するデザインパターンメトリクス

名称	定義
NUMBER(t, k)	時刻 t で適用されている，名前 k であるデザインパターンインスタンスの数
ADDED(t, k)	時刻 t で追加され，名前 k であるデザインパターンインスタンスの数
DELETED(t, k)	時刻 t で削除され，名前 k であるデザインパターンインスタンスの数
KIND(t)	時刻 t で適用されているデザインパターンの種類の数
AGE(p)	デザインパターンインスタンス p がファイル中に現れていた時間

この際，デザインパターンそのものの品質や，デザインパターンがソフトウェアに与える影響を分析する際には，何らかのメトリクスを用いるのが有用である．ソフトウェアの進化に着目したメトリクスとしては，「追加コード行数」や「ある時点で同時に修正されたファイル数」といった，変更メトリクスが挙げられる [24, 66]．そこで，本研究では，変更メトリクスをもとにデザインパターンに関する変更メトリクス（デザインパターンメトリクス）を提案する．

表 7.4 に提案するデザインパターンメトリクスを示す．この中で，NUMBER や KIND を観察することで，対象ソフトウェアの各時点においてどのようなデザインパターンが適用されているかを明らかにすることができる．また，ADDED や DELETED を観察することで，ソフトウェア進化の過程において，どのようなパターンが適用されているか，また取り除かれていくか，を明らかにすることができる．さらに AGE を観察することで，ソフトウェアが進化する過程において，どのようなパターンが安定して用いられているか，といったことが明らかになると考えられる．

7.5. ケーススタディ

先に述べた手法を用いた分析として，本研究ではデザインパターンの推移を観察することで，デザインパターンの追加・削除とクラス数の変動との関連について分析を行った．

7.5.1. 実験概要

分析にあたっては，以下の 2 つの仮説を立てた．

仮説 1 デザインパターンの追加・削除が多く行われ，クラスの追加・削除があまり行われていない場合は，リファクタリングが行われている．

仮説 2 クラスの追加・削除が多く行われ，デザインパターンの追加・削除があまり行われていない場合は，リファクタリング以外の機能の追加・削除・修正作業が行われている．

これらの仮説を検証するため，オープンソースソフトウェアに対して提案手法を適用した．実験対象として，JUnit[3] を採用した．JUnit を選択した理由は，以下の通りである．

1. ソースコードが公開されている．また，ソースコードはソースコード管理システムを用いて管理されている．
2. Java で開発されており，多くのリビジョンにおいてビルドが成功した．
3. 多くの開発者，ユーザによって細粒度の履歴が記録されている．

分析を行うにあたって，SOURCEFORGE.NET[4] で公開されている CVS リポジトリを取得し，cvs2svn[1] を用いて Subversion リポジトリ形式に変換を行った[*1]．実験では，取得した JUnit のリポジトリを Subversion リポジトリ形式に変換した後，チェンジセットごとにチェックアウトを行い，チェックアウトしたリポジトリ全体に対してパターン検出を行った．

[*1] JUnit は SVN 形式のリポジトリも公開されているが，実質的に利用されておらず，CVS 形式のリポジトリが利用されている

7.5.2. 結果

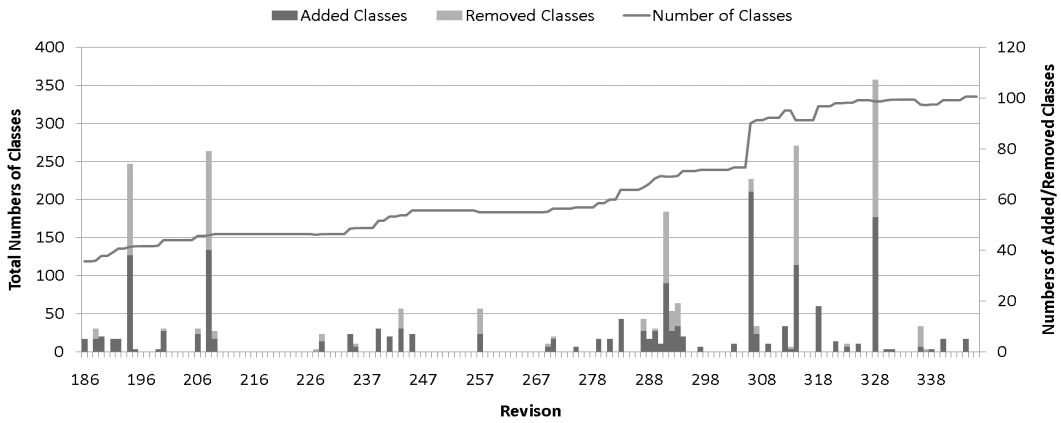

図 7.1　クラス数の遷移

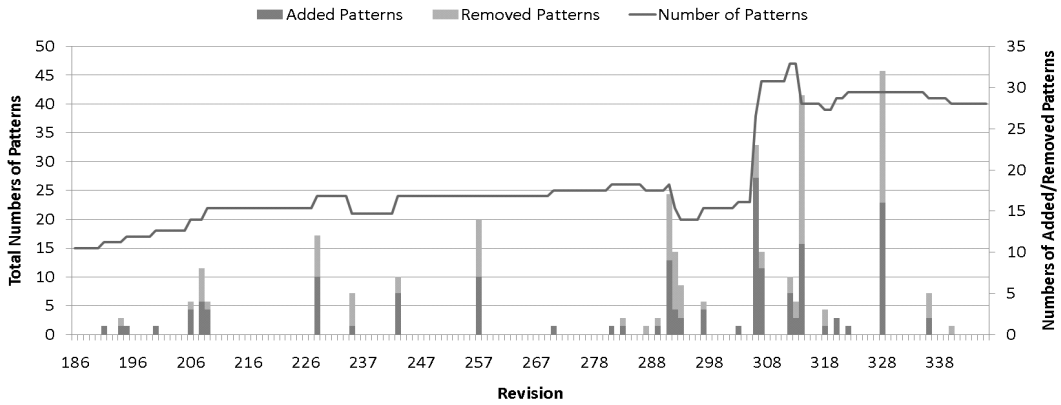

図 7.2　パターン数の遷移

各リビジョンにおけるクラス数およびパターン数の推移を，それぞれ図 7.1，7.2 に示す図 7.1，7.2 に示した分析期間は，リビジョン 186 から 348 までである．そのうち，リビジョン 234 と 277 はビルドが失敗したために除いている．この中で特に追加，削除されたクラス，パターンの数が多かったリビジョンに着目した．修正されたクラスの数またはパターンの数が多かったリビジョン上位 10 件を抽出した．ここで修正されたクラスの数とは，追加されたクラスの数と削除されたクラスの数の和

表 7.5 実験結果

リビジョン	パターン		クラス		仮説を支持するか？		修正/リファクタリング
	修正数	順位	修正数	順位	仮説 1	仮説 2	
194	2	19	74	4		○	Enhancement
208	8	8	79	3		○	Enhancement
243	7	9	17	9	○	○	Refactoring, enhancement
257	14	6	17	9	○		Refactoring
291	17	4	55	6	○		Refactoring
292	16	5	16	11	○		Refactoring
293	6	12	19	7			Refactoring
306	23	3	68	5	○	○	Refactoring, enhancement
307	10	7	10	14	○		Refactoring
312	7	9	10	14	○		Refactoring
314	29	2	81	2	○	○	Refactoring, enhancement
318	3	18	18	8		○	Fix
328	32	1	107	1	○		Refactoring

* 修正数は，追加，削除されたパターン数/クラス数の総和.

である．同様に，修正されたパターンの数とは，追加および削除されたパターンの数の和である．この上位 10 件について，Subversion のコミットログおよび，対象リビジョンでの修正箇所を分析し，該当リビジョンでの修正が，リファクタリング作業に依るものなのか，それ以外の機能追加やバグ修正による修正であるかを判定した．分析を行ったリビジョンおよび該当リビジョンでのクラス数，パターン数，および修正内容の概要を表 7.5 に示す．

7.5.3. 考察

表 7.5 から，JUnit プロジェクトにおいては仮説が支持された．表 7.5 より，10 件中 9 件が仮説 1 を，10 件中 6 件が仮説 2 を支持した．

仮説 1 に関して，リファクタリングを行う場合には，クラスの追加・削除を行わず，クラス内部の構造を大幅に変更することがある．特に，リファクタリング時にデザインパターンを利用する場合には，パターン適用に必要な最小限のクラスを追加する程度である．このように，リファクタリングの場合には，クラス数の増減があまり見られなくても，デザインパターンの数が大幅に変動すると考えられる．

一方，仮説 2 に関して，機能追加を行う場合には，新たな機能に相当するクラスを追加したり，内部クラスとして実装したりするため，クラス数が増える．また機能修正を行う場合は，クラスの削除や内部クラスの追加を行うため，同様にクラス数が増える傾向にある．このため，クラス数の増減が大きく，パターン数の増減があまり見られない場合は機能追加・修正であることが多いと考えられる．

今回の実験では，クラスの追加・削除数およびデザインパターンの追加・削除数が共に高い値を示したリビジョンがあった．このようなリビジョンに関して分析を行ったところ，多くの場合，ソフトウェアのパッケージ構成を変更したため，クラスの追加削除およびパターンの追加削除が共に大幅に発生したことが確認された．これは，今回実験で利用した cvs2svn が，パッケージ構成の変更（CVS 上ではディレクトリの削除およびファイルの追加として扱われる）を，Subversion でも同じくディレクトリの削除およびファイルの追加として扱ったためと思われる．Subversion でこのような操作を行う場合には，ディレクトリ名の変更もしくはファイルの移動として扱う．そのため，実際に Subversion で適切に管理されたリポジトリの場合は，クラスの追加・削除数はあまり変動せず，パターン数が大幅に変動すると考えられる．

7.6. 関連研究

開発履歴を用いたソフトウェアの解析に関しては，多くの研究がある．履歴情報を用いた，ソフトウェアのソースコード中に含まれる重複コード（コードクローン）の分析手法がいくつか提案されている．Kim らは，クローンセットの生存期間を分析することでクローンの性質を探求することを目的としている [49]．川口らは，個々のクローンセットを分析することで，修正が必要な箇所を開発者に提示することを目指している [46]．また，類似の分析として，関数を対象とする履歴追跡手法 [21, 22] や，クラス単位での履歴追跡手法 [5] が提案されている．

版管理システムを利用した研究としては，版管理システムに存在する開発履歴から関連性の高い関数を提示する研究 [15, 80] や，バグ修正支援 [77]，行数の変化や開発者ごとの編集を行った行数などを図示する試み [14, 17] が行われている．本研究では，これらの手法と比較して，設計工程や保守工程での影響を大きく受けると考えられる，デザインパターンに着目し，履歴情報を用いた分析を行っている．

7.7. おわりに

本研究では，版管理システムを用いた，デザインパターンの適用過程を分析する手法を提案した．また，提案手法を利用した分析例として，実際にオープンソースソフトウェアに適用し，デザインパターンとソフトウェアのクラス数との関連について分析を行った．本手法を用いることで，ソフトウェアにおいてデザインパターンがどのように利用されているかを定量的に明らかにすることができる．これにより，デザインパターンの利用が開発プロダクトに与える影響を分析することが可能となる．

今後の展望としては，CK メトリクス [12] などのオブジェクト指向メトリクスや品質評価手法と組み合わせ，デザインパターンが開発プロダクトに与える影響についての調査が挙げられる．また，個々のデザインパターンがソフトウェアの開発中にどのようなライフサイクルをたどるかについての分析も挙げられる．

第 8 章

結論

本研究では，ソフトウェア開発における定量的管理の支援を目的として，(A) 定量的管理に用いる定量データの測定計画立案フレームワークの構築，および (B) 開発プロセスの定量的評価・分析手法の確立を行った．

まず，組織で策定された管理指標を利用した組織的な定量的管理を実施している国内のあるソフトウェア開発組織を対象に，定量的管理がどのように実施されているかを確認した．その結果，多くの管理指標が有効に活用されている一方で，プロジェクトの状況によっては有効に活用されていない場合や，プロジェクトの実態に合わせて調整を行っている上で指標を利用していることが判明した．管理指標の調整にあたっては，指標利用に必要な定量データの収集に関して，定量データの粒度と測定頻度の観点から調整を行っていることが明らかになった．調査の結果からは，開発組織における定量的管理の実態とともに，定量的管理を実施する上での課題が明らかとなった．具体的には，定量的管理計画立案に関する課題と，開発プロセスの定量的評価・分析に関する課題である．

定量的管理計画立案に関する課題としては，計画立案作業の手順が不明瞭であること，管理内容を調整する視点が未整備であることが挙げられる．そこで，調査結果に基づき定量的管理計画の策定，調整作業を体系的に行うためのフレームワークを構築した．提案フレームワークは，定量的管理計画の立案に必要な組織レベルおよびプロジェクトレベルの作業手順を系統的に整理したものである．また，提案フレームワークをもとにして，定量的管理計画の立案を支援するシステム AQUAMarine を

開発した．AQUAMarine は国内のソフトウェア開発組織にて，その有効性を確認するため評価を行った．その結果，提案フレームワークおよび AQUAMarine が定量的管理計画の立案作業に対し，有効な支援機能を提供していることを確認した．これより，本研究で提案した定量的管理計画立案支援のためのフレームワークおよびシステムは，実開発プロジェクトの管理計画策定において，有効に機能することが期待される．

開発プロセスの定量的評価・分析に関する課題としては，プロセスに関するデータのみを収集することが難しいこと，定量的な評価基準が明確でないことが挙げられる．本研究では，自動的に収集することが可能な定量データを用いた開発プロセスの評価手法を検討した．具体的には，あらかじめ定められた開発プロセスの遵守度，プロセス中に発生した手戻りの状況などを開発プロセスの評価尺度とし，開発プロセスとそのプロセスを通して開発されたソフトウェアとの関連を分析する手法を提案した．提案手法をオープンソースソフトウェア開発プロジェクトに適用した結果，開発プロセスの評価尺度を抽出することが可能であることを確認した．また，ソフトウェアの設計が，開発を通してどのように反映されたか分析するために，ソフトウェア中に含まれるデザインパターンに着目した，ソフトウェア設計の時系列分析手法も提案した．提案手法を実際の開発プロジェクトデータに適用した結果，開発プロセス中の，ソフトウェアの設計品質を向上させる作業が行われている箇所を特定することが可能であることを確認した．

これら本論文の成果により，ソフトウェア開発プロジェクトの定量的管理計画を迅速に策定することが可能となる．また，作成した計画に基づいて収集した定量データを用いて，開発プロセスを定量的に評価，改善して行くことが可能となる．これにより，従来と比較して，より短いサイクルで効果的に定量的管理を実施でき，ソフトウェア開発プロセスの評価，改善を行うことが期待される．

謝辞

本研究を進めるにあたり，多くの方々に御指導，御協力，御支援頂きました．ここに謝意を添えて御名前を記させて頂きます．本当にありがとうございました．

奈良先端科学技術大学院大学 情報科学研究科 飯田 元 教授には，本研究の全過程において熱心な御指導を賜りました．研究方針だけではなく，研究に対する姿勢，研究者としての心構え，論文執筆，発表方法についても多くの御助言を頂きました．また，企業との共同研究という貴重な機会を提供してくださったことは，研究生活において非常に有益なものでした．5 年間大学院で充実した生活を送ることができたのは，先生の御指導と御人柄によるものと確信しております．心より厚く御礼を申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 西谷 紘一 教授 には，本研究を進めるにあたり，貴重な御指導を賜りました．学内の発表において多数の御質問と御指導を頂きました．先生から頂いた御指摘は，研究を客観的に見つめ直す上で重要なものとなりました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 松本 健一 教授には，要所要所で本研究に対し貴重な御指導，御助言を賜りました．また，企業との共同研究に際しても，御助言を頂きました．先生の厳しくも温かい御指導のおかげで，研究者としての考え方を身につけることができたと確信しております．心より感謝申し上げます．

大阪大学大学院 情報科学研究科 楠本 真二 教授には，本論文に対し貴重な御助言，御指摘を賜りました．また，ワークショップで御一緒させて頂いた際にも，研究方針に対して温かい御助言を賜りました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 門田 暁人 准教授には，本論文に対し貴重な御助言を賜りました．また，研究活動や学生生活に関しても，常日頃から

様々な御助言を頂きました．心より感謝申し上げます．

株式会社 日立製作所 福地 豊 様，ならびに 米光 哲哉 様には，貴重な企業資料を御提供頂きました．また，論文投稿の際には，実務の観点から重要な御意見を数多く頂戴しました．心より感謝申し上げます．

管理指標の利用実態調査に御協力頂いた皆様，ならびに AQUAMarine の評価レビューに御協力頂いた皆様には，業務で御忙しい中，貴重な御時間を頂戴し，研究を進めるに当たって貴重な情報を御提供頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 川口 真司 助教には，本研究を進めるにあたり，広範囲かつ多大な御助力を頂きました．特に，学会発表や論文投稿時に貴重な御助言を頂戴いたしました．また，研究や教育，学生生活に関しても貴重な御助言を賜りました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 名倉 正剛 特任助教（現 株式会社 日立製作所）には，研究活動や学生生活に関する貴重な御助言を賜りました．特に論文投稿時や発表時には非常に熱心に御指導頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 大平 雅雄 助教，森崎 修司 助教には，研究に対する姿勢から，研究生活の楽しさ，厳しさなど，数多くのことを学ばせて頂きました．心より感謝申し上げます．

神戸大学大学院 工学研究科 中村 匡秀 准教授には，先生が本学に在籍されていらっしゃった頃から，研究課題の設定，研究論文の書き方や発表作法など，研究者に必要な素養について御指導を賜りました．また，研究指導や授業など，教育に関しても多数の御助言を頂戴しました．心より厚く御礼申し上げます．

株式会社 日立製作所 松井 貴元 様には，国際会議やワークショップで御会いした縁もあり，その後様々な機会を通して貴重な御意見を頂戴しました．特に，研究成果の実務への適用という観点からの御意見は，研究を見直す上で重要なものとなりました．心より厚く御礼申し上げます．

神戸大学大学院 工学研究科 井垣 宏 特命助教，京都産業大学 コンピュータ理工学部 玉田 春昭 助教，南山大学 数理情報学部 中道 上 講師，ソニー株式会社 岡本 圭司 博士，大阪大学大学院 情報科学研究科 柿元 健 特任助教，奈良工業高等専門学校 情報工学科 上野 秀剛 助教には，皆様が本学に在籍されている頃から研究との関係の有無に拘わらず，様々な相談に乗って頂き，多くのアドバイスを頂きました．心よ

り感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 角田 雅照 特任助教，戸田 航史 博士，柗本 真佑 氏には，研究を進めるにあたって様々な相談に乗って頂き，貴重なアドバイスを多数頂戴しました．また，研究以外でも大変親しくして頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 石田 響子 女史（現 株式会社 日立製作所），伊原 彰紀 氏，高田 純 氏（現 マイクロソフト株式会社）には，分析用ツールの開発やデータ収集など，研究を進める上で多数の御助力を頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア設計学講座，ならびにソフトウェア工学講座の皆様には，日頃より多大な御協力と御助言を頂き，公私ともに支えて頂きました．5 年間充実した生活を過ごせたのは皆様のおかげです．特に同期生の大蔵 君治 氏，後藤 慶多 氏（現 日本電子計算株式会社），亀井 靖高 博士，瀧進也 氏（現 株式会社 日立製作所），田中 章弘 氏（現 株式会社 日立製作所），三井 康平 氏（現 株式会社インターネットイニシアティブ）には，常日頃より楽しい時間を過ごさせて頂きました．ありがとうございました．

最後に，日頃より私を励まし，応援してくださった両親に心より深く感謝します．

参考文献

[1] cvs2svn. http://cvs2svn.tigris.org/.

[2] Design pattern detection tool. http://java.uom.gr/~nikos/pattern-detection.html.

[3] JUnit. http://www.junit.org/.

[4] SourceForge.net. http://sourceforge.net/.

[5] Giuliano Antoniol, Massimiliano Di Penta, and Ettore Merlo. An automatic approach to identify class evolution discontinuities. In *Proceedings of the 7th International Workshop on Principles of Software Evolution (IWPSE'04)*, pp. 31–40, Kyoto, Japan, September 2004.

[6] Zsolt Balanyi and Rudolf Ferenc. Mining design patterns from C++ source code. In *Proceedings of the 19th IEEE International Conference on Software Maintenance (ICSM'03)*, p. 305, September 2003.

[7] Victor R. Basili, Lionel C. Briand, and Walcélio L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transaction on Software Engineering*, Vol. 22, No. 10, pp. 751–761, October 1996.

[8] Victor R. Basili and H. Dieter Rombach. Tailoring the software process to project goals and environments. In *Proceedings of the 9th International Conference on Software Engineering(ICSE '87)*, pp. 345–357, March 1987.

[9] Victor R. Basili and David M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transaction on Software Engineering*, Vol. 10, No. 3, pp. 728–738, November 1984.

[10] Ulrike Becker-Kornstaedt, Dirk Hamann, Ralf Kempkens, Peter Rösch, Mar-

tin Verlage, Richard Webby, and Jörg Zettel. Support for the process engineer: The spearmint approach to software process definition and process guidance. In *Proceedings of the 11th International Conference on Advanced Information Systems Engineering*, pp. 119–133, June 1999.

[11] Barry Boehm and Dan Port. Conceptual modeling challenges for model-based architecting and software engineering (MBASE). *Lecture Notes in Computer Science*, Vol. 1565, pp. 24–43, 1999.

[12] S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, pp. 476–493, June 1994.

[13] CMMI Product Team. Capability maturity model integration for system engineering / software engineering / integrated product and process development, version 1.1. Technical Report CMU/SEI-2002-TR-004, Software Engineering Institute, August 2002.

[14] Christian Collberg, Stephen Kobourov, Jasvir Nagra, Jacob Pitts, and Kevin Wampler. A system for graph-based visualization of the evolution of software. In *Proc. ACM symposium on Software visualization (SOFTVIS 2003)*, pp. 77–86, San Diego, California, USA, June 2003.

[15] Davor Cubranic, Gail C. Murphy, Janice Singer, and Kellogg S. Booth. Hipikat: A project memory for software development. *IEEE Transactions on Software Engineering*, Vol. 31, No. 6, pp. 446–465, June 2005.

[16] Michael Cusumano, A MacCormack, Chris F Kemerer, and B Crandall. Software development worldwide: the state of the practice. *IEEE Software*, Vol. 20, No. 6, pp. 28 – 34, November 2003.

[17] S. G. Eick, T. L. Graves, A. F. Karr, A. Mockus, and P. Schuster. Visualizing software changes. *IEEE Transactions on Software Engineering*, Vol. 28, No. 4, pp. 396–412, April 2002.

[18] Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Professional, 1999.

[19] 伏田享平, 大前勇輝, 名倉正剛, 川口真司, 大蔵君治, 飯田元. バグ管理システムを対象としたアジャイルソフトウェア開発における保守プロセスの観察. 電子情報通信学会技術報告, ソフトウェアサイエンス研究会, 第 SS2008-39 巻, pp. 1–6, November 2008.

[20] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[21] Michael W. Godfrey and Lijie Zou. Using origin analysis to detect merging and splitting of source code entities. *IEEE Transactions on Software Engineering*, Vol. 31, No. 2, pp. 166–181, February 2005.

[22] Nicolas Gold and Andrew Mohan. A framework for understanding conceptual changes in evolving source code. In *Proceedings of the 19th International Conference on Software Maintenance (ICSM 2003)*, pp. 22–26, September 2003.

[23] Anandasivam Gopal, Tridas Mukhopadhyay, and M. S. Krishnan. The impact of institutional forces on software metrics programs. *IEEE Transaction on Software Engineering*, Vol. 31, No. 8, pp. 679–694, August 2005.

[24] Todd L. Graves, Alan F. Karr, J. S. Marron, and Harvey Siy. Predicting fault incidence using software change history. *IEEE Transaction on Software Engineering*, Vol. 26, No. 7, pp. 653–661, July 2000.

[25] Jennifer Gremba and Chuck Myers. The ideal model: A practical guide for improvement. *Software Engineering Institute (SEI) publication, Bridge*, No. 3, 1997. `http://www.sei.cmu.edu/ideal/ideal.bridge.html`.

[26] Tracy Hall and Norman Fenton. Implementing effective software metrics programs. *IEEE Software*, Vol. 14, No. 2, pp. 55–65, 1997.

[27] 花川典子. ソースコードの複雑さに影響を与えるソフトウェア開発マイクロプロセス抽出方法の提案. 飯田元, 山本里枝子（編）, ソフトウェアエンジニアリング最前線 2008, ソフトウェアエンジニアリングシンポジウム 2008（SES2008）, pp. 95–102, September 2008.

[28] Dirk Heuzeroth, Thomas Holl, Gustav Hogstrom, and Welf Lowe. Auto-

matic design pattern detection. In *Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC '03)*, pp. 94–105, Los Alamitos, CA, USA, May 2003. IEEE Computer Society.

[29] 本田紘介, 土肥正, 岡村寛之. モジュールサイズの分布に基づいたソフトウェア欠陥密度の評価に関する考察. 電子情報通信学会論文誌 A, Vol. 86, No. 6, pp. 713–717, June 2003.

[30] Watts S. Humphrey. The personal software process. Technical Report CMU/SEI-2000-TR-022, Software Engineering Institute, November 2000.

[31] Ming Huo, He Zhang, and Ross Jeffery. A systematic approach to process enactment analysis as input to software process improvement or tailoring. In *Proceedings of the 13rd Asia Pacific Software Engineering Conference (APSEC'06)*, pp. 401–410, December 2006.

[32] 伊原彰紀, 大平雅雄, 松本健一. 障害管理システム利用時の修正遅延要因の分析. ソフトウェア信頼性研究会 第 5 回ワークショップ論文集, pp. 75–82, March 2009.

[33] International Organization for Standardization. *ISO/IEC 12207:1995, Information Technology – Software Life Cycle Processes*. International Organization for Standardization, Geneva, Switzerland, 1995.

[34] International Organization for Standardization. *ISO/IEC 9126-1:2001, Software engineering – Product quality – Part 1: Quality model*. International Organization for Standardization, Geneva, Switzerland, June 2001.

[35] International Organization for Standardization. *ISO/IEC 15939:2002, Software Engineering – Software Measurement Process*. International Organization for Standardization, Geneva, Switzerland, 2002.

[36] International Organization for Standardization. *ISO/IEC 15504-2:2003, Information Technology – Process assesment – Part 2: Performing an assesment*. International Organization for Standardization, Geneva, Switzerland, October 2003.

[37] International Organization for Standardization. *ISO/IEC TR 9126-2:2003, Software engineering – Product quality – Part 2: External metrics*. Interna-

tional Organization for Standardization, Geneva, Switzerland, July 2003.

[38] International Organization for Standardization. *ISO/IEC TR 9126-3:2003, Software engineering – Product quality – Part 3: Internal metrics*. International Organization for Standardization, Geneva, Switzerland, July 2003.

[39] International Organization for Standardization. *ISO/IEC 15504-1:2004, Information Technology – Process assesment – Part 1: Concepts and vocabulary*. International Organization for Standardization, Geneva, Switzerland, November 2004.

[40] International Organization for Standardization. *ISO/IEC 15504-3:2004, Information Technology – Process assesment – Part 3: Guidance on performing an assesment*. International Organization for Standardization, Geneva, Switzerland, January 2004.

[41] International Organization for Standardization. *ISO/IEC 15504-4:2004, Information Technology – Process assesment – Part 4: Guidance on use for process improvement and process capability determination*. International Organization for Standardization, Geneva, Switzerland, July 2004.

[42] International Organization for Standardization. *ISO/IEC TR 9126-4:2004, Software engineering – Product quality – Part 4: Quality in use metrics*. International Organization for Standardization, Geneva, Switzerland, March 2004.

[43] International Organization for Standardization. *ISO/IEC 15504-5:2006, Information Technology – Process assesment – Part 5: An exemplar Process Assessment Model*. International Organization for Standardization, Geneva, Switzerland, March 2006.

[44] 石田響子, 伏田享平, 名倉正剛, 川口真司, 飯田元. 「非障害案件」の発見を目的としたバグ票の観察. 情報処理学会研究報告, 第 2008 巻, pp. 1–8, September 2008.

[45] 岩切博, 大曽根一将, 藤原良一, 中前雅之. 高度化プロセスにおける定量的プロジェクト管理の実践と効果. *SEC journal*, Vol. 2, No. 4, pp. 42–49, November 2006.

[46] 川口真司, 松下誠, 井上克郎. 版管理システムを用いたクローン履歴分析手法の提案. 電子情報通信学会論文誌 D, Vol. J89-D, No. 10, pp. 2279–2287, December 2006.

[47] Marc I. Kellner, Ulrike Becker-Kornstaedt, William E. Riddle, Jennifer Tomal, and Martin Verlage. Process guides: Effective guidance for process participants. In *Proceedings of the 5th International Conference on the Software Process*, pp. 11–25, June 1998.

[48] Joshua Kerievsky. *Refactoring to Patterns*. Pearson Higher Education, 2004.

[49] Miryung Kim and David Notkin. Using a clone genealogy extractor for understanding and supporting evolution of code clones. In *Proc. 2nd Intl. Workshop on Mining Software Repositories (MSR 2005)*, pp. 17–21, Saint Louis, Missouri, May 2005.

[50] Felicia Kurniawati and Ross Jeffery. The use and effects of an electronic process guide and experience repository: a longitudinal study. *Information and Software Technology*, Vol. 48, No. 7, pp. 566–577, July 2006.

[51] Oliver Laitenberger and Jean-Marc DeBaud. An encompassing life cycle centric survey of software inspection. *The Journal of Systems and Software*, Vol. 50, No. 1, pp. 5–31, 2000.

[52] John McGarry, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Fred Hall. *Practical Software Measurement: Objective Information for Decision Makers*. Addison Wesley Professional, October 2001.

[53] Philip M.Johnson, Hongbing Kou, Joy Agustin, Christopher Chan, Carleton Moore, Jitender Miglani, Shenyan Zhen, and William E. J. Doane. Beyond the personal software process: metrics collection and analysis for the differently disciplined. In *Proceedings of the 25th International Conference on Software Engineering (ICSE2003)*, pp. 641–646, May 2003.

[54] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology*, Vol. 11, No. 3, pp. 309–346, 2002.

[55] Nils Brede Moe and Tore Dybå. The use of an electronic process guide in a medium-sized software development company. *Software Process: Improvement and Practice*, Vol. 11, No. 1, pp. 21–34, January 2006.

[56] 門田暁人, 佐藤慎一, 神谷年洋, 松本健一. コードクローンに基づくレガシーソフトウェアの品質の分析. 情報処理学会論文誌, Vol. 44, No. 8, pp. 2178–2188, August 2003.

[57] Shuji Morisaki, Tomoko Matsumura, Kimiharu Ohkura, Kyohei Fushida, Shinji Kawaguchi, and Hajimu Iida. Fine-grained software process analysis to ongoing distributed software development. In *Proceedings of the 1st Workshop on Measurement-based Cockpits for Distributed Software and Systems Engineering Projects (SOFTPIT 2007)*, pp. 26–30, August 2007. Munich, Germany.

[58] Raimund Moser, Witold Pedrycz, and Giancarlo Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Proceedings of the 30th international conference on Software engineering (ICSE '08)*, pp. 181–190, New York, NY, USA, 2008. ACM.

[59] Jürgen Münch and Jens Heidrich. Software project control centers: concepts and approaches. *Journal of Systems and Software*, Vol. 70, No. 1-2, pp. 3 – 19, 2004.

[60] 中村建助, 矢口竜太郎. 2008 年情報化実態調査 プロジェクトの成功率は 31.1%. 日経コンピュータ 2008 年 12 月 1 日号, No. 718, pp. 38–53, December 2008.

[61] 日本規格協会. JIS X 0160:1996, ソフトウェアライフサイクルプロセス. 日本規格協会, 1996. 2001 確認.

[62] 日本規格協会. JIS X 0141:2004, ソフトウェア測定プロセス. 日本規格協会, 2004.

[63] 小笠原秀人, 古賀恵子, 木田昌平, 澤口邦夫, 高橋幸司, 南方良紀, 重本一郎. ソフトウェアメトリクスの有効性評価. 日科技連 第 20 年度 (2004 年度) 分科会成果報告, 2005. `http://www.juse.or.jp/software/study_data2004_1.html`.

[64] 小笠原潤, 片野寿昭, 中山貴史, 吉村宏之. 訓練によるソフトウェアプロセス改善の基盤作り. 飯田元, 山本里枝子（編）, ソフトウェアエンジニアリング最前

線 2008，ソフトウェアエンジニアリングシンポジウム 2008（SES2008）, pp. 87–94, September 2008.

[65] 大平雅雄, 横森励士, 阪井誠, 岩村聡, 小野英治, 新海平, 横川智教. ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム. 電子情報通信学会論文誌 D-I, Vol. J88-D-I, No. 2, pp. 228–239, February 2005.

[66] Thomas J. Ostrand and Elaine J. Weyuker. Predicting the location and number of faults in large software systems. *IEEE Transaction on Software Engineering.*, Vol. 31, No. 4, pp. 340–355, April 2005.

[67] Soojin Park, Hoyoung Na, Sooyong Park, and Vijayan Sugumaran. A semi-automated filtering technique for software process tailoring using neural network. *Expert Systems with Applications*, Vol. 30, No. 2, pp. 179–189, February 2006.

[68] Shari Lawrence Pfleeger, Ross Jeffery, Bill Curtis, and Barbara Kitchenham. Status report on software measurement. *IEEE Software*, Vol. 14, No. 2, pp. 33–43, 1997.

[69] Project Management Institute. プロジェクトマネジメント知識体系ガイド 第 3 版. Project Management Institute, 2005.

[70] F. Riguzzi. A Survey of Software Metrics. Technical report, 1996.

[71] 高橋良英. C 言語ソフトウェア保守工程における halstead のソフトウェアサイエンス計測と障害密度との関係の分析. 電子情報通信学会論文誌. D-I, Vol. 82, No. 8, pp. 1017–1034, August 1999.

[72] Gregory Tassey. *The Economic Impacts of Inadequate Infrastructure for Software Testing.* Diane Pub Co, 2003.

[73] The Eclipse Foundation and Eclipse Process Framework Project. Eclipse process framework composer. `http://www.eclipse.org/epf/`.

[74] Nikolaos Tsantalis, Alexander Chatzigeorgiou, George Stephanides, and Spyros T. Halkidis. Design pattern detection using similarity scoring. *IEEE Transactions on Software Engineering*, Vol. 32, No. 11, pp. 896–909, 2006.

[75] 角田雅照, 玉田春昭, 森崎修司, 松村知子, 黒崎章, 松本健一. コードレビュー指摘密度を用いたソフトウェア欠陥密度予測. 情報処理学会論文誌, Vol. 50,

No. 3, pp. 1144–1155, March 2009.

[76] University of Southern California and Carnegie Mellon University. MBASE 577 interactive process guide. `http://sunset.usc.edu/research/MBASE/EPG/home.html`.

[77] Chadd C. Willams and Jeffrey K. Hollingsworth. Automatic mining of source code repositories to improve bug finding techniques. *IEEE Transactions on Software Engineering*, Vol. 31, No. 6, pp. 466–480, June 2005.

[78] Walter Zimmer. *Relationships between Design Patterns*, pp. 345–364. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[79] Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. Predicting defects for eclipse. In *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, May 2007.

[80] Thomas Zimmermann, Peter Weissgerber, Stephan Diehl, and Andreas Zeller. Mining version histories to guide software changes. *IEEE Transactions on Software Engineering*, Vol. 31, No. 6, pp. 429–445, June 2005.

[81] 独立行政法人 情報処理推進機構ソフトウェア・エンジニアリング・センター. IT プロジェクトの「見える化」 上流工程編. 日経 BP, 2007.

[82] 独立行政法人 情報処理推進機構ソフトウェア・エンジニアリング・センター. ソフトウェア開発データ白書 2008. 日経 BP, 2008.

[83] 経済産業省 情報処理振興課, 社団法人 日本情報システム・ユーザー協会. ソフトウェアメトリックス調査 2008. 社団法人 日本情報システム・ユーザー協会, 2008.

付録

A. ヒアリングで利用したアンケートの詳細

第 3 章で実施したヒアリングで利用した質問用紙を図 A.1 から図 A.3 に示す.

管理指標評価のための基礎データ収集を目的としたアンケート
「開発プロジェクトにおける管理指標利用の実態調査」

【調査の目的】
本アンケートはその一環として，御社で策定・推奨されているプロセス関連の管理指標 44 個に対して，プロジェクト種別ごとの利用実態調査を行うものです．お忙しいところ貴重なお時間をいただき誠に恐縮ですが，本調査にご協力くださいますよう，お願い申し上げます．収集されたデータは，下記の目的に利用いたします：

- プロジェクトの業種や規模ごとに各指標の利用実績を把握する
- 初級管理者によるプロジェクト計画時に，業種や規模に応じた管理指標を推薦する
- 社内で閲覧可能な事例データベース（固有名詞その他の情報はマスクします）構築の材料とする

【データの機密保持および利用について】
本アンケートにて収集したプロジェクト固有の情報は上記の目的にのみ利用し，御社および奈良先端科学技術大学院大学以外の第三者に提供することは一切ありません．収集した情報は漏洩することのないよう、万全の体制により厳重に管理いたします。ただし、統計的数値等の抽象的な分析結果につきましては，御社ご担当者様から事前の了解を得た上で論文や外部発表などに使用させていただくことがあります。

なお，ご提供いただいた回答は、御社ご担当者様によって十分に精査を行っていただき、不適切と判断された情報は事前にマスクした状態で NAIST に引き渡していただきますので、説明文等を附記いただく際には，できるだけ具体的な表現を用いていただきますようご配慮をお願いいたします。

以下，ご自身の関わられたプロジェクトについて，以下の質問にお答えください．
※複数プロジェクトについてご回答いただく場合は誠に恐縮ですが，個々のプロジェクト毎に解答用紙一式を用いてお答え願います．

セクション A．プロジェクトのプロファイル

A-1. プロジェクト名を記入し、対象業務を〔〕の中に記入してください：
プロジェクト名※（　　）　対象業務〔　　〕
※プロジェクト名は追加質問等を行う際の追跡用に御社内部のみにて利用いたします。NAIST には伝わりませんので、よろしくご記入をお願いいたします。

A-2. プロジェクトの現在の状況を〔〕の中からから選び○で囲んでください：〔 既に完了　現在進行中　その他（　　）〕

A-3. プロジェクトにおけるあなたの立場に最も近いものを〔〕の中からから選び○で囲んでください：〔 PM　QA　PMO　その他（　　）〕

A-4. 当該プロジェクトまでのご自身の経験について、その経験年数および経験プロジェクト数をお答えください：
経験年数：（　　）年、うち、A-3 で回答した役割での経験年数：（　　）年
これまでに管理したプロジェクトの概数（およそ：　　）個

A-5. 当該プロジェクトの規模を書き入れてください、また単位を〔〕の中からから選び○で囲んでください。
（　　）単位〔 ステップ　FP　その他（　　）〕

図 A.1　ヒアリングで利用した質問用紙（プロジェクトのプロファイル）

セクションB. 各管理指標の利用実態

各指標の利用実態に関する以下の項目について、最も近いと思われるものを選択してください（次ページ以降のリスト中にご記入ください）。なお、管理指標の名称および各選択項目は略称を記載しております。管理指標の詳細につきましては別紙資料1「管理指標定義表」をご参照願います.
評価項目は，管理指標に関する提供情報，あなたの理解度，運用レベル，適用効果の 4 分野に分かれます. 各質問および選択項目の意味は下記の通りです。

提供情報内容の評価に関する質問:

・B-1. **目的の重要度**:各指標には，それぞれに設定された管理目的が存在しますが，それが今回のプロジェクト管理にどの程度重要であると思われますか？5 段階での評価をお願いします（最も重要な場合を5としてください、わからない場合には 1 でお答えください）。

・B-2. **目的と指標定義の対応**:各指標に設定された目的と指標自体の定義（利用データや解釈の仕方）が適切に対応していると思われますか:
- −適切＝目的と指標が合致している
- −不適切＝指標の定義は必ずしも目的と合致していない（※合致しない点を具体的にお書きください）
- −わからない

・B-3. **記述のわかりやすさ**:添付された指標定義に関する説明はどの程度わかりやすくかかれていると思われますか(ご自身の理解度とは別に評価ください)
- −理解しやすい
- −理解しにくい（※わかりにくい点を具体的にお書きください）

理解の度合いの評価に関する質問:

・B-4. **目的の理解度**:あなたはこの指標に設定された目的を理解していると考えますか
- −はい＝十分に理解しているとおもう
- −いいえ＝十分には理解していないと思う

・B-5. **指標定義の理解度**:あなたはこの指標の原理（必要となる測定データや分析法，運用方法）をどの程度理解していると考えますか
- −はい＝十分に理解しているとおもう
- −いいえ＝十分には理解していないと思う

・B-6. **利用経験**:あなたはこれまでにこの指標を実際に利用した経験がありますか
- −ある＝当該プロジェクト以前利用したことがある
- −ない＝これまで利用したことがない

指標の利用方式に関する質問:

・B-7. **指標利用の方式**:今回のプロジェクトでは、この指標はどのように利用しましたか
- −指示通り＝「管理指標定義表」に書かれているとおりに利用した
- −調整有り＝プロジェクトの実情に応じた調整をしたうえで運用した（※具体的な調整内容をお書きください）
- −不採用＝利用せず（※不採用の理由を具体的にお書きください）

・B-8. **データ収集の方式**:指標に必要なデータの収集体制について（指標自体が不採用の場合は回答不要です）
- −指示通り＝「管理指標定義表」に書かれているとおりの方法で収集した
- −調整有り＝プロジェクトの実情に応じた方法で収集した（※具体的な調整内容をお書きください）
- −不十分＝とりあえず形式的にデータを収集したが適切な計測を行ったとはいえない（※具体的な理由があればお書きください）

指標の利用効果に関する質問

・ B-9. 効果の有無:今回のプロジェクトにおいて、この指標は対応する管理目的の達成に役立ったと思いますか. 5 段階での評価をお願いします（非常に役立った場合ー＝5、役に立たなかった、わからない場合＝1 でお答えください）:

・ B-10. 利用効果に対する問題点:B-9 で、この指標が有効でなかったと思われる場合にはその原因を選んでください
- −目的不一致:目的に対して指標の定義は不適切であった
- −運用上の理由:利用しなかったもしくは運用上の不備などがあった
- −それ以外の理由（※具体的にお書きください）

図 A.2　ヒアリングで利用した質問用紙（各管理指標の利用実態）

セクションC. 総合的な質問

C-1:**プロジェクト独自の管理指標**:B に挙げた指標以外で、プロジェクト独自で利用した(定義した)指標があれば、どのような目的でその指標を定義し、利用するためにどのような基礎データを収集したかをお答えください。

(　　)

C-2: **追加すべき指標:**その他, セクション B に挙げた指標群に欠けていると思われる管理目的や指標があれば具体的に指摘をお願いします

(　　)

C-3: **管理指標利用効果向上のための施策:** 管理指標群利用効果の向上にはどのような方策が必要と思いますか. 必要と思われるものに 5 段階で重要度を記入してください(5が最も重要)

- ■ 指標の定義に関するガイド　( )
- ■ 分析・管理のトレーニング　( )
- ■ 計測のトレーニング　( )
- ■ 自動化された計測環境　( )
- ■ 自動化された分析環境　( )
- ■ プロジェクトに応じて指標をカスタマイズする手法　( )
- ■ 適用事例集　( )
- ■ その他(具体的に:)
  - 内容〔　　〕　( )
  - 内容〔　　〕　( )
  - 内容〔　　〕　( )
  - 内容〔　　〕　( )
  - 内容〔　　〕　( )

質問は以上です. ありがとうございました

図 A.3　ヒアリングで利用した質問用紙（総合的な質問）

B. 3.5 節で取り上げた指標の詳細

表 B.1 から表 B.5 に，3.5 節で言及した管理指標の詳細を示す.

表 B.1　管理指標の詳細（進捗管理）

指標番号	名称	目的	必要な測定データ	測定方法
4	日程計画の変更状況	スケジュールをレビューし，課題を分析し是正処置をとる手続きが適切に行われているかどうかを確認し，スケジュールを維持する	1. 日程計画の変更回数 2. 日程計画の変更回数のうち，適切な理由が記されている回数	1. 社内定例進捗会議にて収集 2. PMO が適切さを判断し，計測・収集する

表 B.2　管理指標の詳細（レビュー）

指標番号	名称	目的	必要な測定データ	測定方法
22	規模当たりのピアレビュー時間と，レビュー時に発見された欠陥数の関係	レビューでより多くの欠陥を摘出するための良い条件（レビュー時のレビュー速度）を求める	1. レビュー時に発見された欠陥数 2. レビュー対象規模 3. レビュー時のレビュー時間 4. レビューのレビューア人数	各レビュー時にプロジェクトリーダに測定データが報告される
23	規模当りのレビューア個々の準備時間と，事前指摘件数との関係	レビューでより多くの欠陥を摘出するための良い条件を求める	1. レビュー前のレビューア個々の準備時間 2. レビュー対象規模 3. レビュー前のレビューア個々の指摘件数	各レビュー時にプロジェクトリーダに測定データが報告される
24	レビュー時に指摘された欠陥数と，次工程以降におけるレビュー済み工程を起因とする欠陥数との関係	レビューデータを分析し，レビューの効果を測定する．	1. 原因となった工程毎の欠陥数 2. レビューの対象成果物の欠陥数／コード行数	1. 各工程のレビュー時に，レビューの対象成果物の作成者が，得られた欠陥がどの工程に起因した欠陥数であるかを測定 2. レビューの対象成果物の作成担当者が，欠陥数／コード行数を記録
25	レビューカバー率と，次工程以降におけるレビュー済み成果物を起因とする欠陥数との関係	レビューデータを分析し，レビューの効果を測定する．	1. レビューカバー率・チェック項目のカバー率 2. 次工程以降におけるレビュー実施工程を起因とする欠陥数／コード行数	1. 各成果物の担当者がレビュー終了時に，レビューカバー率を測定する． 2. サブリーダがプロジェクト終了時にレビュー実施成果物の仕様・コードを起因とする規模あたりの欠陥数を集計する．
26	システムの機能毎の原因別欠陥数	レビューデータを分析する（種類別欠陥数を調べ，的確な対策を講じる）	レビューの際の，原因別の欠陥数	レビューの際に，担当者が原因別の欠陥数をカウントして，サブリーダに報告する．
27	レビュー時に指摘された問題の未解決件数と，期限遅れの未解決件数の推移	各種レビューにおいて，指摘件数に対する是正状況を把握，フォローする．	1. レビュー時の指摘の未解決件数 2. 期限遅れの未解決件数	指摘事項の解決担当者がサブリーダに報告する．

表 B.3 管理指標の詳細（プロセス品質保証）

指標番号	名称	目的	必要な測定データ	測定方法
33	プロセス QA 検査の実施率，稼働後の事故件数，原価（実績）／原価（予定），および試送後の重大不良数	プロセス QA 検査の実施状況とその効果を測定する	1. プロセス QA 監査の実施率 2. 事故発生件数 3. 原価（実績）／原価（予定） 4. 試送後の重大不良数	プロジェクトの完了後に，1.2. からプロセス QA 監査の効果を測定する．
34	プロセス QA 検査のチェックリストのチェック率，稼働後の事故件数，原価（実績）／原価（予定），および試送後の重大不良数	プロセス QA 検査の実施状況とその効果を測定する	1. プロセス QA 検査のチェックリストのチェック率 2. 事故発生件数 3. 原価（実績）／原価（予定） 4. 試送後の重大不良数	プロジェクトの完了後に，1.2. からプロセス QA 監査の効果を測定する．
35	作業改善指示書のカテゴリ毎の是正率と，要因別事故発生件数	プロセス QA 検査の実施状況とその効果を測定する	1. カテゴリ毎の是正処置済み件数／指摘件数 2. 要因別不良発生件数	プロジェクトの完了後に，1.2. からプロセス QA 監査の効果を測定する．
36	作業改善指示書の是正処置済み件数と，是正措置未済件数	プロセス QA が適切に実施されているかどうかを監視する．問題点の解決状況を最後まで追跡する	1. 作業改善指示書 (プロセス QA 検査用) の是正処置済み件数 (累計) 2. 作業改善指示書 (プロセス QA 検査用) の是正処置未済件数	プロジェクトリーダが未済件数が 0 になるまで毎週収集

表 B.4 管理指標の詳細（リスク管理）

指標番号	名称	目的	必要な測定データ	測定方法
38	リスク要因毎の発生確率	実際に起こったリスクを分析し，組織のリスク管理のプロセス改善に役立てる．	1. リスク要因毎の発生件数 2. チェックリストでリスク管理を行っているプロジェクト件数	1. 月毎にリスク要因毎の発生件数を報告する 2. 期毎にチェックリストでリスク管理を行っているプロジェクト件数を測定する

表 B.5　管理指標の詳細（支援プロセス）

指標番号	名称	目的	必要な測定データ	測定方法
44	契約内示から取引先に対する内示迄の期間，およびプロジェクト着手日から契約内示日迄の期間	調達が所定の手続きに沿って実施されているかどうかを監視する．	1. 契約内示日 2. 取引先に対する内示日 3. プロジェクト着手日 4. プロジェクト着手日から契約内示迄の期間 5. 契約内示日から取引先に対する内示日迄の期間	修正中
45	測定項目毎の測定率	測定データ報告率の推移を監視し，測定プロセスの実施状況を把握し，フォローする．	1. 部門毎のプロジェクトの測定率平均 2. プロジェクト毎の測定率 3. プロジェクト毎にサブシステムの測定率を平均したもの 4. 測定項目毎の測定率 5. 報告回数 6. 計画された測定回数	月次でサブリーダが測定項目毎の測定率＝報告回数／計画された測定回数を求める．
46	ベースライン毎で計画された構成管理対象成果物数の保管件数	構成管理が適正に実施されているかどうかを把握する．	ベースライン毎で計画された構成管理対象成果物数の保管件数	サブリーダがプロジェクト計画時にサブシステムで構成管理の対象とする成果物を決め，報告する．
47	各フェーズに求められる構成監査の実施回数と目標回数	構成監査が実施されているかを確認する．	1．PM用としてサブシステム毎に構成監査回数 2. システム開発部門長のために，プロジェクト毎にサブシステムの実施回数の平均 3. 支援部門のために，システム開発部門毎の実施回数の平均	フェーズ毎