博士論文

ソフトウェア開発プロジェクトの変動マネジメント手法

――形式知化と知識継承――

大島　丈史

主指導教員　内平 直志

北陸先端科学技術大学院大学
先端科学技術研究科［知識科学］

令和元年 **9** 月

Abstract

As the importance of IT system has been increased, how to share the knowledge of project management is important. In a software development project, software size, progress and productivity tend to differ from the project plan for various reasons. However, it is difficult to know the precise varying status of the project quantitatively, because the real situation of the software development project cannot be grasped easily. If the granularity of the grasped progress and cost are rough and not accurate, project manager cannot find the delay of progress until the delay expand, and the control of the project might fail.

To solve these problems, this study focuses on the knowledge transfer for the management of project variance. First, it proposes the method for the variance of software development project, which paternize the past project knowledge and systemization of variance visualization. Next, it proposes the knowledge stuructured model to generalize the knowledge structure based on the project variance management method. This method enables the smooth project management by utilizing the past project knowledge. And the knowledge structure model will contribute the organizational knowledge externalization and knowledge transfer of project management.

The key success facor for the software project management is to garasp the variance of the project early and take the appropriate action to prevent the increase of the variance according to the cause of the variance by utilizaing past knowledge. EVM (Earned Value Management) method is one of the quantitatie management method which can manage the schedule and cost systematically. However, main usage of EMV has been to grasp the status of the project monthly or weekly. It is difficult to grasp the precise and detailed WBS related progress and effort because it takes much effort to collect the data and to analize detailed information. This study proposes the method to make the variance of the project visible quantitatively and daily to solve these problems. This method makes the variance visible every day by the unit of function and person by systemizing data collection and automating progress calculation using man-hour and software size by the unit of detailed WBS. It has been shown through a case study that this method helps project manager to control the project smoothly by finding problems early, analyzing the causes of the problem, and forecasting the cost.

Next, this study proposes the knowledge structure model as the framework for the externalization of the knowledge, by generalizing and systemizing the knowledge element regarding the project variance management. In ISO21500 and PMBOK, knowledge is classified by the project phase and category of the management process. They can be used for grouping the knowledge for project management, but they don’t provide the concrete knowledge structure and the relations between the knowledge from the viewpoint of monitoring or control of the project. The knowledge model is structured with knowledge layer by modeling the project management from the viewpoint of controlling the project variance, and by defining the dynamic relation between each knowledge layer. As for the project management knowledge, there are the easy one to make explicit and the difficult one. The knowledge classification model is proposed for software development. It is shown that the knowledge structure model contributes to the knowledge externalization and knowledge transfer in the organizations.

Keywords : Quantitative Project Management, Variance of Progress and Cost, EVM, Knowledge Externalization, Knowledge Transfer

概要

社会や経済における IT システムの重要性が増しており、プロジェクトマネジメントの知識をいかに共有するかが重要な課題となっている。ソフトウェア開発プロジェクトでは、規模、進捗、コスト等が計画に対して変動しやすいことに加え、それらの変動状況を適時に正確に把握することが難しい。進捗やコストの管理の粒度が粗く不正確な場合は、異常の察知が遅れ、原因究明やコントロールに支障をきたすことがある。プロジェクトマネジャーは過去のプロジェクトの経験を元にして問題が発生しやすい点等に着目して重点的な対策を行うが、様々な変動への対処方法は、プロジェクトマネジャーの暗黙知となっている場合が多く、過去の経験に基づくノウハウの活用や組織内での知識継承は難しい。

本研究では、このような課題に対して、プロジェクトの変動への対処に関する知識の活用や継承に焦点を当て、詳細な粒度での変動の可視化を自動化する手法と知識の形式知化を組み合わせて過去の知識を活用するための変動マネジメント手法と、マネジメントに必要な知識を体系化した知識の構造モデルを提案する。これにより、過去の知識を活用したマネジメントの円滑な実施を可能とするとともに、組織内でのプロジェクトマネジメントの知識の形式知化や継承に貢献する。

ソフトウェア開発のマネジメントについては、定量的管理に関する先行研究が多数行われているが、計画の精度を高めるためのメトリクスや見積りの精度向上に視点が置かれていた。本研究では、プロジェクトの進展に伴って発生する変動のマネジメントへの対処に焦点を当てる。プロジェクトマネジメントにおいては、変動を短いサイクルで把握し、早期に変動の原因に応じた対策を講じ変動を抑制することが重要であり、それを実現する方法と、対処に必要な過去の知識の活用が成功の鍵となる。変動把握の手法の一つとして、進捗とコストを統合的に管理する EVM (Earned Value Management) 手法があるが、その活用方法はプロジェクト全体の傾向を把握するための俯瞰的な活用に止まり、日々の変動把握や変動の原因分析等への活用は難しかった。そこで、WBS の単位を機能別、人別の数時間単位の詳細な粒度とすることで、EVM を日次での詳細な管理粒度での変動を可視化する手法へ拡張し、早期の問題発見や問題の原因究明及びコントロールへの活用を可能とした。この際に、管理粒度の詳細化に伴う運用負荷を抑制するため、工数計画の作成や進捗率の計上を自動化する仕組みを実現した。変動に対するコントロールについては、個々のプロジェクトマネジャーの経験に基づくノウハウとなっていた工数の再見積りや、原因に応じた対処方法等をパターン化することで形式知化し、過去の経験やノウハウの活用によるコントロールを可能とした。本手法により、早期の問題把握と、原因に応じた適切な対処が可能となり、スケジュール遅延等の問題を未然に防止するための円滑なプロジェクトマネジメントが可能となることを、プロジェクトへの適用によって検証した。

また、プロジェクトマネジメントの知識体系としては、ISO21500 や PMBOK などのプロ

ジェクトマネジメントの知識体系がある。これらはマネジメントプロセスの種類で知識を分類したものであり、網羅的な知識の整理には適するが、過去のプロジェクト経験などの具体的な知識の形式知化や活用の枠組みとして活用する上では、知識間の関連性等の情報が不足している。本研究では、プロジェクトを制御対象として捉えたときの視点から知識を構造化するとともに、監視・制御のプロセスと知識領域間の関係を整理した知識構造モデルを提案する。これにより、状況に応じた知識の活用や、形式知化及びシステム化が可能な知識等を識別、分類することが容易となる。この知識構造モデルによって、具体的な知識の活用促進や、形式化が必要な対象領域の選定、及び知識領域に応じた適切な形式知化の方法の検討等が可能となることを、適用方法の例示及び考察によって示した。

目次

図表目次

1. 図

2. 表

1. 研究の目的と論文の構成

1.1. 研究の背景

1.1.1. ソフトウェア開発のプロジェクトマネジメントの知識継承上の課題

社会や経済の発展における IT システムの重要性が増す中で、プロジェクトマネジメントの知識をいかに共有するかが、各組織におけるプロジェクトを成功に導くために重要となっている。日経コンピュータの「IT プロジェクトの実態調査 2018 調査」（日経 BP 社 2018）によると、1,745 件のシステム導入または刷新プロジェクトのうち、47.2％が「失敗」と回答されている。

各組織ではプロジェクトの成功率向上のため、プロジェクトマネジメントの知識継承を目的として、過去のプロジェクト経験を元にしたガイドや教訓集の作成、事例研修会やケーススタディによる研修などの、様々な知識継承の取り組みを行っている。しかし、なかなかその効果が現れてこないケースも多い。この背景には、プロジェクトマネジャーに必要な知識が多岐に渡ることや、知識の多くが個人の暗黙知となっていることなどが背景にあると考えられる。

したがって、このように人への依存性が高く、暗黙知が占める部分の多いプロジェクトマネジメントの知識を、他のプロジェクトマネジャーが活用できるよう、各組織内でいかに継承するかがプロジェクトの成功のための重要な課題であると言える。

1.1.2. ソフトウェア開発の変動の定量的マネジメントの課題

このようなソフトウェア開発のプロジェクトマネジメントにおける課題に対して、定量的管理手法を確立することによって過去のデータや経験を知識として活用可能とするため、ソフトウェア規模や工数の見積り手法、成果物の品質分析手法、進捗管理手法等に関する先行研究が行われてきた。

ただし、ソフトウェア開発の成果物はドキュメントやソースコード等であり、開発作業には成果物としては現れない検討作業やレビュー等の様々な作業が含まれるため、規模や進捗、作業の状態等の正確な見積りや把握が難しく、規模や工数の見積りには誤差が含まれることが避けられない。さらに、仕様の追加に伴う規模増加や、人に依存する生産性の変動等の様々な要因によって、計画に対して規模や進捗、コスト等の変動が発生する。これらの規模、進捗、コストの変動の把握が遅くなると対処が遅れ、把握の精度が不正確であると、適切な対処ができずに問題が拡大するなどのリスクがある。

また、変動に対するコントロールは、プロジェクトマネジャーの経験に基づく暗黙知に依存する部分が大きい。例えば、様々な原因によって発生する進捗遅延やコスト増加等の

問題への対処方法に関する知識は、プロジェクトマネジャーの過去の経験に基づく暗黙知に依存する部分が大きく、組織内での知識継承が難しい。

したがって、ソフトウェア開発プロジェクトの変動に関するマネジメントにおいては、規模、進捗、コスト（工数）等の変動を、精度良く短いサイクルで定量的に把握してコントロールするためのマネジメント手法と、効果的なマネジメントのために、暗黙知を含む過去のプロジェクトの教訓などの知識をいかに蓄積し活用するかが重要な課題である。

なお、本研究で扱う変動とは、プロジェクト計画を立案した際に前提としていた規模、生産性、スケジュール、コスト等が、プロジェクトの進行とともに増減することを意味する。

1.2. 本研究の目的

本研究では、前述のソフトウェア開発における課題に対して、プロジェクトマネジメントの知識の活用や継承をテーマとし、特に難しい変動マネジメントの知識を形式知化して活用するためのマネジメント手法を提案する。合わせて、変動マネジメントを中心とした知識要素の構造化及び体系化を行なうことで、ソフトウェア開発のプロジェクトマネジメントに関する知識の形式知化やシステム化を行う際の枠組みとして活用可能な知識構造モデルを提案する。

初めに、過去のプロジェクトで蓄積された知識や方法を活用して変動に対する監視や制御を円滑に行うためのマネジメント手法を提案する。具体的には、日々の変動を定量的に計測し、可視化することによってプロジェクトを制御するための一連のマネジメントプロセスをモデル化しシステム化する手法を提案する。この際、進捗や工数の変動を毎日、詳細な作業の単位で把握しようとすると運用負荷が高く実行が難しいという問題に対して、規模や生産性などのパラメータから工数計画を自動的に作成する方法や、実績工数データの情報から進捗率を自動的に計上する仕組みによって解決を図る。この手法によって、ソフトウェア規模の変動や進捗、コスト等の変動を可視化し、異常の早期察知、原因分析及び予測、ステークホルダ間での合意形成や計画変更等を円滑に進めることが可能となる。また変動への対応方法については、個々のプロジェクトマネジャーの暗黙知となっている仕様調整、スケジュール調整及びそのための要員数や工数の再見積方式を含むコンロールに関する知識をパターン化することによって形式知化し、定量的マネジメントプロセスに埋め込むことで、知識の継承や活用を促進することを狙いとする。

さらに変動の可視化プロセスや知識のパターン化に必要な知識を抽出し、管理要素間の関連性を整理することによって、具体的な知識の形式知化や活用を目的とした知識構造モデルを提案する。このモデルの活用方法の一つとして、知識をシステム化や形式知化が可能な部分とそれ以外の暗黙知等の要素に分類するための方法を、利用可能データとの関係性の観点で体系的に整理する。これによって、知識の領域や特性に応じた効果的な知識継

承方法や、プロジェクトマネジメントの知識をシステム化等によって補完するための研究の発展に貢献する。

本研究のリサーチクエスチョンを以下のように設定した。

【メジャーリサーチクエスチョン（MRQ）】

ソフトウェア開発プロジェクトにおいて、プロジェクトマネジャーは各種変動をどのように定量的に把握し、いかなる知識を活用することによってプロジェクトを計画通りにコントロールすることが可能となるのか。また、この際に活用される知識はいかなる要素から構成され、知識の間にはどのような関係性があるのか。

【サブシディアリリサーチクエスチョン（SRQ）】

SRQ1：プロジェクトマネジャーは、各種の変動をいかなる方法によって負荷低減と精度向上を両立しながら把握することが可能となるのか。

SRQ2：プロジェクトマネジャーは、変動に対していかなる情報や知識をどのように活用することでプロジェクトを円滑にコントロールすることが可能となるのか。

SRQ3：プロジェクトの変動に対処するための知識は、いかなる要素から構成され、各要素はどのような関係性を有するのか。

1.3. 研究の意義

プロジェクトの定量的マネジメントに関する先行研究においては、定量化の手法としてのメトリクスの精度向上や、見積り及び計画の制度を向上させるための研究が主体であったのに対し、本研究では先行研究における各種メトリリクスを、計画作成、変動可視化、原因分析等のマネジメントに活用する方法に焦点を当てたものである。プロジェクトの規模、進捗、コスト等の変動の可視化手法と合わせて、プロジェクトマネジメントの知識をシステム化や形式知化によって活用、継承するための手法を提案する。先行研究で提案されている各種メトリクスはこの際の変動可視化や原因分析に活用する。本手法により、過去のプロジェクトの実績データの活用方法等を含む人間のノウハウを形式知化し、プロジェクトマネジメントの知識の継承を促進することが可能となる。

知識科学の観点では、従来の研究が、知識継承のプロセスとしてのモデル化の観点や、知識の性質による分類方法等が研究対象であったのに対して、本研究では、プロジェクトマネジメントの知識継承の手段としての知識の形式知化を進める上での、知識の構成要素と知識要素間の関連性を明らかにするためのモデルを提案する。この際、プロジェクトを制御対象として捉え、プロジェクトの監視及び制御のプロセスと対応させて知識要素を構造化し、モデル化する。これによって、マネジメントのプロセスに対応した具体的な知識

の整理や、継承すべき知識の特定、及び対象領域に応じた形式知化の検討を行う際の枠組みを提供し、知識継承方法の研究に貢献するものである。

1.4. 本論文で扱う基本用語の定義

1.4.1 プロジェクトマネジメントの用語

本論文で扱うプロジェクトマネジメントに関する用語について定義しておく (PMI 2017) 。

(1) プロジェクト

ある目的を達成するために行う計画の策定およびそれを遂行すること。有期性と独自性、不確実性を特徴とする。

(2) プロジェクトマネジメント

プロジェクトを成功に導くために行う活動。プロジェクトの目標を達成するための観点として、PMBOK (PMI 2017) では、スコープ、時間、コスト、品質、人的資源、コミュニケーション、リスク、調達、統合管理などの観点を知識エリアとして定義している。

(3) 変動

本研究で扱う変動とは、プロジェクト計画を立案した際に前提としていた規模、生産性、スケジュール、コスト、品質等が、プロジェクトの進行とともに増減することを意味する。

(4) 可視化

物理的に見えにくいソフトウェアやITシステムの進捗、生産性、品質等をステークホルダの間で共有できるよう、データや文書によって状況を把握できるようにすること（（木野 2011）、(IPA 2011) ）。

(5) コントロール

PMBOKやISO21500で定義しているプロセスグループの一つであり、プロジェクトの進捗などを定常的に監視、測定することによって計画との差異を特定し、是正措置などを行うことによってプロジェクトの目的を達成するようにすることである。

1.4.2 知識継承に関する用語

本研究ではプロジェクトの変動に対するマネジメントを中心とした知識継承とそのための知識の形式知化に焦点を当てたものである。以下にDavenport et al. (1998)及び内平（2010）による知識に関する用語の定義を示す。

(1) 知識

知識は、知識の所有者の中で、所有者の価値観、過去の経験、課題・問題意識、現在

の状況認識と結びついているもので、新しい情報に対して所有者の解釈・判断・行動を生み出すものである。

(2) 知識継承

同一組織内での異なる世代間の知識移転や、組織的な知識の継続的な保持を意味する。「知識移転」とは、送り手の頭の中にある知識を受け手の頭の中に再構築することである。ここで世代間には、先行プロジェクトと後続プロジェクトの間も含まれる。知識継承は、同じ組織における知識移転を含んでいるが、知識移転に加えて組織的な人材育成や知識管理基板の整備等も含まれる (De Long 2004)。

本稿では、プロジェクトマネジメントの知識を組織的に蓄積し、異なるプロジェクトのプロジェクトマネジャーに移転するための知識の形式知化や分類に焦点を当てる。

(3) 形式知と暗黙知

知識継承について論じる際には、形式知と暗黙知という分類が重要な意味を持つ。暗黙知という概念は、Polanyi (1980) によって”Tacit Knowing”として提示されたものであり、組織の一部となっていて文書化されていない知識であり、エキスパートの技術や勘、ノウハウなどである。形式知は、文章にできる知識であり、作業マニュアルや説明書などである。暗黙知は形式知に比べ捉えにくく、学習が困難であるとされている。

Nonaka & Takeuchi (1995) は、形式知と暗黙知を、企業内の知識継承の文脈で用いており、SECIモデルと呼ばれる、形式知と暗黙知の間でのスパイラルな知識継承のモデルを提案している。

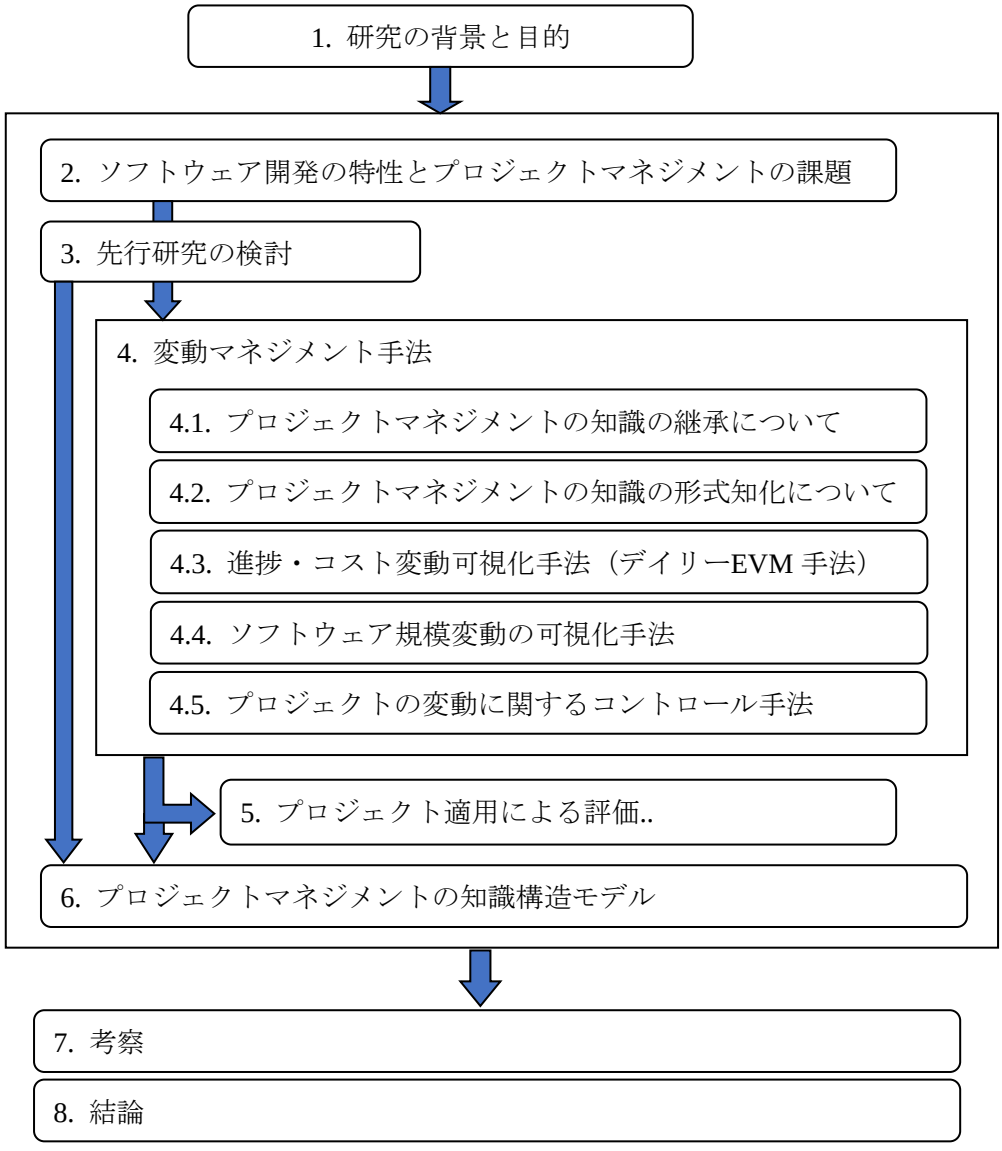
1.5. 本研究の方法と構成

本研究では、研究戦略としてデザインサイエンスの方法論 (Hevner et al. 2004, Van Aken 2005) を用いることによって、従来の課題を解決するための手法の提案と、適用事例に基づく手法の有効性を示す。先行研究レビュー（第3章）に続いて、先行研究において不足していたプロジェクトの変動マネジメントの課題を解決するための手法（可視化及びコントロール手法）の提案を行い（4章）、プロジェクトへの適用例を通じて効果を示す（第5章）。変動マネジメント手法については、出来高（進捗）とコストを一体とした定量化によるプロジェクトのパフォーマンスの評価等のための活用が中心であったEVMを、毎日の変動を可視化しコントロールに活用する手法へと拡張する新たなマネジメント方法（デイリーEVM手法）を提案する。この際、実用上の阻害要因となる運用負荷を低減するための工数計画作成、データ収集と進捗率計上を自動化する仕組みを取り入れる。さらに、進捗や工数を把握する際の基礎となるソフトウェア規模の変動可視化手法を組み合わせた方法として提案する。次に、変動に関するコントロールの手法を提案し、規模、進捗、コスト（工数）の変動可視化からコントロールまでの一連のマネジメントプロセスの全体像を示す。本手法の適用効果について、プロジェクトへの適用事例の考察によって示す（第5章）。

次に、変動マネジメント手法を適用する際に必要となる知識を構造化、体系化し、知識構造モデルの提案とモデルの適用方法の考察を行う（第6章）。

以上の変動マネジメント手法と知識構造モデルについて、先行研究との関係、応用範囲、課題等の観点から考察し（第7章）、最後に結論として、リサーチクエスチョンへの回答、本研究の適用範囲と限界、及び今後の展望を述べる（第8章）。

本論文の構成は以下の通りである。



2. ソフトウェア開発のプロジェクトマネジメントの特性と課題

2.1. ソフトウェア開発のプロジェクトマネジメントの現状

ITの適用領域は年々拡大し続け、近年ではネットワークとスマートフォンや各種の機器を繋げたIoTやクラウド環境の利用拡大等に伴い、ITが広く深く社会に浸透しつつある。新たなビジネスモデルの創出等の手段としてもITは重要な位置づけとなっており、システムやソフトウェアの位置付けは益々重要性を増している。これに伴い、ソフトウェア開発の難易度も高くなっており、ソフトウェア開発のプロジェクトマネジメントを的確に進めることができる人材の育成や知識の継承が重要な課題となっている。

しかし、ソフトウェア開発は人の作業に依存する割合が高く、目に見えないためにソフトウェア規模、スケジュールやコスト等が計画に対して変動することが多く、状況の把握やコントロールが難しい。さらに、プロジェクトマネジメントの知識は、プロジェクトマネジャーが経験によって習得する要素や暗黙知となっていることが多く、知識の継承が難しい。このため、ソフトウェア開発におけるプロジェクトマネジメントの方法は、プロジェクトや個人によって異なっており、有効なマネジメント手法が組織内で共有されにくい。

以下にソフトウェア開発プロジェクトの特性と課題を整理する。

2.2. ソフトウェア開発プロジェクトの特性と知識活用の課題

ソフトウェア開発プロジェクトには、以下のような特性や問題が存在する (Boem 1984) 。

(a) 完成するまで機能を完全に把握することが難しい

(b) 作業の多くが人のスキル、能力、経験に依存する

(c) 定量的に表すことが難しい

(d) コスト見積りの誤差が大きい

これらの特性は、ソフトウェア開発の成果物がドキュメントやソースコード等であり、開発作業の中には、成果物には直接的には現れない検討作業やレビュー等の人のスキルや経験に依存する様々な作業が含まれることなどが背景にある。

(d)の見積り誤差の問題については、Boem (1984) やIPA (2006) が、ソフトウェア開発のコストの見積りは不確定要素が大きいことに言及している。図 2-1に示すように、工程が上流工程であるほど要件が確定していないため、ソフトウェア規模やコストの見積りの誤差が大きい。工程が企画・構想、要件定義、開発と進むにつれて誤差は減少する。

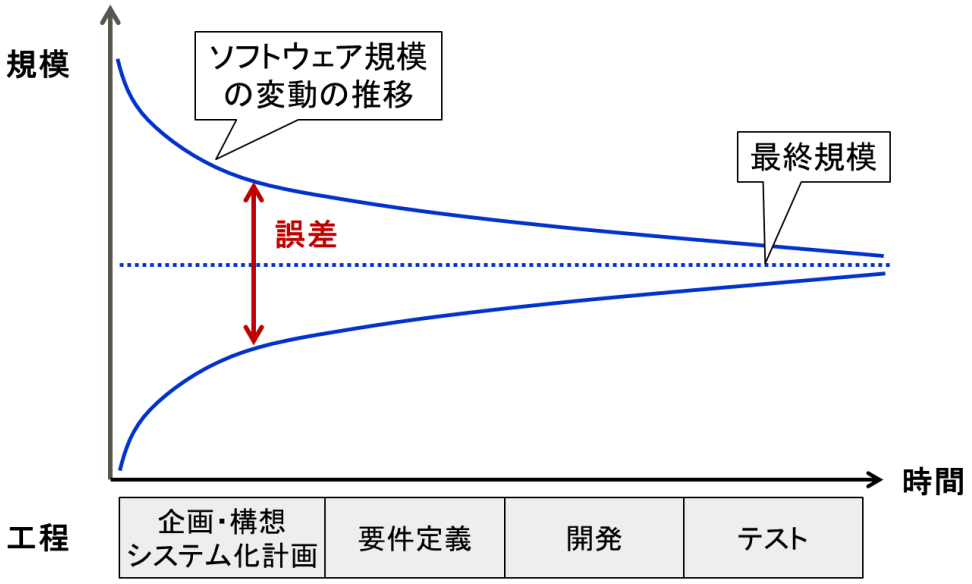


図 2-1　ソフトウェア規模及びコスト見積りの誤差

（出典：Boehm (1981) の“Software Engineering Economics”, Prentice Hall, *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, VOL. SE-10, NO. 1, JANUARY 1984.のFig.3及びIPA (2006) の図3.3を元に、筆者が追記）

このような見積りの誤差が元々含まれていることに加え、仕様の追加に伴う規模増加や人に依存する生産性の変動等の様々な要因によって、計画に対して規模や進捗、コスト等の変動が発生する。これらの規模、進捗、コストの変動の把握が遅くなると対処が遅れ、把握の精度が不正確であると、適切な対処ができずに問題が拡大するなどのリスクがある。

ところが、特性の(a)から(c)に挙げられているように、規模や進捗、作業の状態等の正確な把握が難しい。また、変動に対するコントロールは、プロジェクトマネジャーの経験に基づく暗黙知に依存する部分が大きいのが実態である。したがって、ソフトウェア開発プロジェクトのマネジメントにおいては、常に変動（計画と実績の差異）を精度良く定量的に把握してコントロールしてゆくことが重要となる。この際、暗黙知となっていることが多いコントロールに関する知識をいかに蓄積し活用するかが課題となる。

2.3. プロジェクトの定量的マネジメントに関する課題について

Jones (2008）は、『ソフトウェア開発の定量化手法』の中で、「ソフトウェア開発の定量化は、エンジニアリングのための基礎である。」と述べている。しかし、必ずしもソフトウェア開発に関する各種の定量データを蓄積、活用している組織は多くはないと同書では述べられている。日本国内においても、JUAS (2016）のアンケート調査によると、工程別に何らかの定量的な分析を実施しているプロジェクトは6割となっている。以下に、『ソフト

ウェア開発の定量化手法』に示されている定量的マネジメントに関する主要なテーマとして、「ソフトウェアの規模と工数見積り」、「生産性、品質及び見積り」、「規模の変動と要求、スケジュール」、「工数や時間の記録」に関するテーマを取り上げ、ソフトウェア開発プロジェクトにおけるプロジェクトマネジメントの課題を整理する。

2.3.1. ソフトウェアの規模と工数見積りについて

ソフトウェア開発プロジェクトの計画を立案するためには、開発に必要な工数の見積りが、要員の見積りやスケジュールの策定などの基礎を成す非常に重要なプロセスである。このためには、開発すべきソフトウェアの規模の定量化が第一歩となる。ソフトウェアの規模の見積りや工数の見積りが不正確であると、適切なコントロールが行われないまま、プロジェクトの計画と実績の差異が拡大し、プロジェクトの遅延やコスト増加を招くことに繋がってしまう。

2.3.2. 生産性、品質及び見積りについて

日本国内におけるソフトウェア開発プロジェクトの状況としては、第1章で取り上げたように日経BP社（2018）によると、1,745件のシステム導入／刷新プロジェクトのうち、47.2%が「失敗」と回答されている。また、JUAS (2016) によると、2004年度から2008年度における500人月以上の開発工数のプロジェクトにおける調査結果では、予算を超過したプロジェクトの割合が46.2%、工期を超過したプロジェクトが52.1%である。

米国における状況としては、「2008年時点で、ソフトウェア産業にはいくつかの厄介な問題がある。」と述べ、米国内の調査結果に基づいて、具体例として以下の事実が挙げられている(Jones 2010)。

(a) 国レベルでは、ソフトウェアの生産性は過去20年以上向上していない。
(b) 国レベルでは、ソフトウェア品質レベルでは過去20年間でやや改善しただけである。
(c) 2008年における訴訟に至った問題は基本的に1987年と変わらない。
(d) コスト超過、スケジュール遅延あるいはソフトウェアプロジェクトの全体的失敗は、2008年時点では減少でなく増大している。
(e) ソフトウェア生産性および品質の面では、70%の企業は、向上も低下もせずにそのままの状況に留まっている。
(f) 保守活動はソフトウェア開発活動よりも早い勢いで増え続けている。
(g) ソフトウェア生産性が向上している15%の企業の主たる理由は、アジャイル手法を用いることにより高い生産性が得られているWebアプリケーションの発展にある。

以上のような、ソフトウェア開発の生産性が業界全体として向上していないことや、コスト超過やスケジュール遅延などの問題の発生頻度についても大きい改善はみられないこ

となどが、同書に統計値や調査結果によって示されている。

しかしながら、ソフトウェア開発においては、特に見積りが難しいことや品質の評価が難しいことが大きい課題と認識されていたため、多くの研究は見積りの精度向上や開発された後の品質評価のための研究に重点が置かれてきた。開発の生産性向上のための取り組みとしては、設計情報からのソースコードの自動生成やテスト自動化などの開発方法の改善のための研究が行われてきた。しかし、特定の分野や一部の組織における改善事例等は発表されているが、ソフトウェ業界全体として生産性の向上を示す統計値等はない。

また、開発プロジェクトのコスト超過やスケジュール遅延を防止するための研究としては、過去のプロジェクトマネジメントの失敗経験を生かすための知識の継承やプロジェクトマネジャーの育成に関するものが中心となってきたが、その効果はなかなか現れていない。これは、ソフトウェア開発プロジェクトの規模や進捗、コストの変動が発生しやすいという特性に加え、プロジェクトマネジメントに必要な知識としては、ソフトウェア技術に加え、人のスキルや生産性、作業品質などの特性や組織、文化等を含む多様な知識が必要であり、プロジェクトマネッジャーの経験に基づく暗黙知となっていることが多いためと考えられる。

本研究は、このような問題に対して、プロジェクトの計画と実績の差異を少なくするための変動に対するマネジメントに着目し、変動の定量的マネジメントの方法や、知識の形式知化及びデータ分析のシステム化手法を活用することによって、プロジェクトの円滑なコントロールや、知識の継承に貢献するものである。

2.3.3. 測定及び定量化の現状

ソフトウェア開発の定量化の取り組みとしては、各種のメトリクスを活用するための研究や提案が行われてきた。IPA (2018) の『ソフトウェア開発データ白書 2017-2018』や、JUAS (2018) によるの『ソフトウェアメトリクス調査』等によって、規模、コスト、期間や品質に関する各種の統計データが収集されている。しかし、ソフトウェア開発分野においては、前述のJones (2018) やJUAS (2016) のアンケートに示されているように、定量的なマネジメントを行うために必要な尺度が確立されていると言える状況ではない。プロジェクトの生産性や品質の定量的な予測、管理、制御を目的としたデータの蓄積が行われている組織も少ないことがアンケート結果等に現れている。

本研究では、ソフトウェア規模、工数、進捗等の変動に関するデータを詳細な粒度で毎日蓄積することによって、変動の可視化、分析及びコントロールを行うための手法を提案する。本手法を組織で継続的に適用することによって、プロジェクトの実績データが蓄積され、見積り、予測、制御等のマネジメントの精度を向上させることが可能となる。

2.3.4. 規模の変動と要求、スケジュールについて

ソフトウェアの規模は、開発期間や開発工数の見積り、計画作成及び監視、コントロールの基礎となる重要なメトリクスである。ソフトウェアの規模は要求や仕様の変更などの要因によって変動する。次の点がソフトウェア規模の変動の要因や課題として挙げられている (Jones 2010)。

(a)「アプリケーションの規模と複雑度が増すにつれて、スケジュールの計画と実績の差異は大きく拡大している。」

(b)「徐々に増大するユーザ要求の管理に失敗するなどの誤りは、最長のスケジュールにつながる。失敗への道は無数にあるが、成功の道は限られている。

(c) 「要求のクリープにより、スケジュールは計画から乖離する。」

(d) 「要求のクリープの測定は、コード行数中心の時代には困難であったが、FP値を用いるようになって、その変化率を直接、正確に測定することができるようになった。」

以上のように、要求の増加に伴うソフトウェアの規模の増加によって、スケジュールの計画と実績の差異が大きくなり、プロジェクトの失敗につながることが多い。

また、JUAS (2009) のメトリクス調査によると、プロジェクトの総費用増大の理由として、「開発規模の増加」が最も大きい理由であり、続いて「要件分析作業の分析不十分」、「要求仕様の決定遅れ」等の理由が挙げられている。この開発規模の増大理由としては、次の点が主な理由として挙げられている（複数回答）。

(a) 発注時の仕様詳細検討不足（40.0%）

(b) 見積要求仕様書の不十分さにもとづく仕様増加（35.3%）

(c) 検討時の仕様増加（55.3%）

(d) 発注時と運用開始時期の環境の変化による増加（9.5%）

このような原因で発生する規模の増加は、プロジェクトの費用増加やスケジュール遅延につながるとともに、ステークホルダ間での満足度の低下にもつながる。JUAS (2018) のソフトウェアメトリクス調査（図表6-11-17）によると、仕様変更が大きくなるにつれて、プロジェクト全体の満足度が低下するという傾向が示されている。したがって、規模を定量的に把握し、変動に対しする早期のコントロールが、プロジェクトの成功のためには非常に重要であると言える。

本研究では、進捗及び工数の変動に加え、規模の変動を可視化する手法について提案する。設計工程で定期的に規模の差分を計測するプロセスと、規模と分布を自動的に可視化することによって、規模の変動や規模別の分布状況を可視化し、要求のコントロールや計画の見直し、要員の再配置等に活用することが可能となる。

本研究では、この際に利用する規模計測手法として、FP法と同様に利用者に見える機能を表す外部設計書の情報に基づいて規模を計測するファンクションスケール法（Function

Scale 法。以降、FS法と略す。）を適用した富士通 2013）。FS法を選定した理由は、FP法で必要とするファイルの種類や数が明確になる前の段階であっても、システムの画面レイアウト設計書などの設計情報から機能規模を計測できることと、計測者によらずにぶれのない計測値が得られる点である。ただし、同様の条件を満たす規模計測方法であれば、他の手法でも利用可能である。

2.3.5. 工数や時間の記録について

ソフトウェア開発プロジェクトで行われている進捗やコストの管理方法では、週単位や月単位進捗状況をとりまとめることが一般的である。しかし、Jones (2008) は、以下のように、この方法には問題があると述べている。

(a)「週単位や月単位で工数を測定する場合には、曖昧性が大きい。最も正確なのは時間単位の測定である。」

(b) 「資源追跡と時間記録の欠陥や漏れは、おそらくソフトウェアプロジェクトの測定や見積りにとって最も研究の進んでいない課題であろう。」

(c) 「正確な資源追跡と時間記録について議論する論文や書籍の量は全ての測定や尺度に関する論文の1%にも達しない。」

(a)のように、週単位や月単位での工数測定では進捗が曖昧となることや、遅れに気付くのが遅くなり、対処が遅れることがある。(b)や(c)のように、資源や時間の管理は最も研究が進んでいない課題と述べられている。

本研究で提案する変動マネジメント手法では、従来の方法とは異なり、管理単位の基本となる一つのWBSを、その所要時間が数時間または 1 日以内の詳細な粒度で策定することによって、日々の変動の可視化と、原因分析や早期のコントロールを可能とするものである。この際、工数計画の作成、出来高（進捗）や工数の実績を時間単位で計測することとする。

従来、このような時間単位でのマネジメントの実行が困難であった要因として、計画作成やデータ収集、進捗率の計上等のための運用負荷が高くなってしまうことが挙げられる。そこで、本研究では、計画作成の自動化や進捗率の自動計上を可能とする仕組みによって、運用負荷を増やすことなく、時間単位の変動把握を可能とする方法を提案する。

3. 先行研究の検討

第2章で述べたソフトウェア開発の様々な課題に対して、プロジェクトを定量的にマネジメントするためのメトリクスやプロセス及び知識体系に関する様々な研究が行われてきた。以下にソフトウェア開発のマネジメントプロセス、プロジェクトの定量的マネジメントに必要な各種のメトリクス、工数見積り手法、計画策定手法、進捗管理手法、プロジェクトの変動管理、プロジェクトネジメントの知識継承、システム開発及び研究開発の知識継承のそれぞれの先行研究の状況や成果について述べる。特に、本研究の主たる研究対象であるプロジェクトの変動に対するマネジメントに必要なソフトウェア規模の計測方法や進捗把握方法や、工数見積り手法等を中心に先行研究の概要と課題を整理する。

また、プロジェトマネジメントにおける知識継承に関する先行研究の概要と、本研究の位置付けについて最後に整理する。

3.1. プロジェクトマネジメントの手法及びプロセス

プロジェクトは、ITシステムの開発、建造物の建設、プラント建設、公共事業、研究開発など、様々な業界や業務にわたって進められている。プロジェクトマネジメントは、1942年に行われた米国の原子爆弾の開発「マンハッタンプロジェクト」が始まりとされている。その後、1950年代の「アポロ計画」や弾道ミサイル「ポラリス開発」を通じてプロジェクトマネジメントのプロセスや手法が開発された。具体的な手法については、3.4節の計画策定手法や3.5節の進捗管理手法で述べる。

その後、プロジェクトマネジメントは適用範囲を石油産業や化学産業、航空業界、公共事業、ソフトウェア開発などに適用範囲を広げ、ITの発展に伴う定量的な管理手法などの手法も開発されてきた。定量的管理手法は、3.2節で述べる。

プロジェクトマネジメントに関する知識の体系化や標準化も進められた。1989 年に英国の情報システムのプロジェクトマネジメント標準として英国CCTA (the Central Computer and Telecommunications Agency) が開発したPRINCE (Project in Controlled Environment) が最初の知識体系である。1996年に、PRINCE2 (Project in Controlled Environment, 2nd ver.) が出版された。PRINCE2は、英国におけるプロジェクトマネジメントのデファクトスタンダードとなるとともに英国以外でも活用されている。

米国では、1996年に設立されたPMI (Project Management Institute) によって、プロジェクトマネジメントの知識体系としてPMBOK (Project Management Body of Knowledge) が出版された。現在は第6 版が出版されている (PMI 2017)。

日本においては、1998年から経済産業省とエンジニアリング振興協会のプロジェクトマネジメント導入開発委員会によって、P2M (Program & Project Management) が開発された。これは、日本型のプロジェクトマネジメント知識体系としてまとめられたものである

(PMAJ 2014) 。P2Mでは、イノベーション推進と価値創造に対応できるプログラムやプロジェクトマネジメントへの適用をねらいとしている。

以下に、PMBOKで示されている知識エリアとプロセスグループを示す。

3.1.1. 知識エリア

プロジェクトマネジメントの知識体系の一つであるPMBOK 第6版 (PMI 2017) では、知識エリアとして、下記の10種類に分類している。

(1) 統合マネジメント
(2) スコープマネジメント
(3) タイムマネジメント
(4) コストマネジメント
(5) 品質マネジメント
(6) 人的資源マネジメント
(7) コミュニケーションマネジメント
(8) リスクマネジメント
(9) 調達マネジメント
(10) ステークホルダマネジメント

3.1.2. プロセスグループ

また、PMBOKでは、プロジェクトを遂行するためのプロセスとして、下記の５つのプロセスグループに分類し、その中に４７のプロセスを定義している。

(1) プロジェクトの立上げ
(2) プロジェクトの計画
(3) プロジェクトの実行
(4) プロジェクトの監視・コントロール
(5) プロジェクトの終結

また、PMBOKでは、プロジェクトマネジメントに利用可能な手法やツールを、それぞれの知識エリアとプロセスと結びつけて、提示している。

上記の知識エリアとプロセスについては、ソフトウェア開発プロジェクトにおいても、プロジェクトマネジメントのプロセスや知識を共有するための共有言語として利用されている。以下では、3.1.1.の知識エリアと3.1.2のプロセスグループによって分類されるプロジェトマネジメントの各プロセスを定量的に実行する際に利用されるメトリクスや管理手法に関する先行研究について述べる。

3.2. ソフトウェア開発の定量的マネジメントとメトリクスについて

ソフトウェア開発のプロジェクトマネジメントにおける課題に対して、定量的管理手法を確立することによって過去のデータや経験を知識として活用できるようにするための取り組みとして、ソフトウェア規模や工数の見積手法や進捗管理手法に関する多数の研究が行われてきた。以下では、定量的マネジメントを行う上で重要となる、各種のメトリクスに関する先行研究の状況について述べる。

3.2.1. ソフトウェア開発のメトリクスの種類

ソフトウェア開発を定量的にマネジメントするために、何をどのように定量化すればマネジメントを効果的に行えるかという観点から、様々なメトリクスが提案されてきた。Marciniak (1994) によると、ソフトウェア開発に関するメトリクスは、以下のプロダクトメトリクス、プロセスメトリクス、リソースメトリクスの三種類に大別される。図 3-1に示すように、プロジェクトマネジメントにおいては、これらのメトリクスをモニタリングの手段として活用し、コントロールを行う（阿萬ほか 2011）。

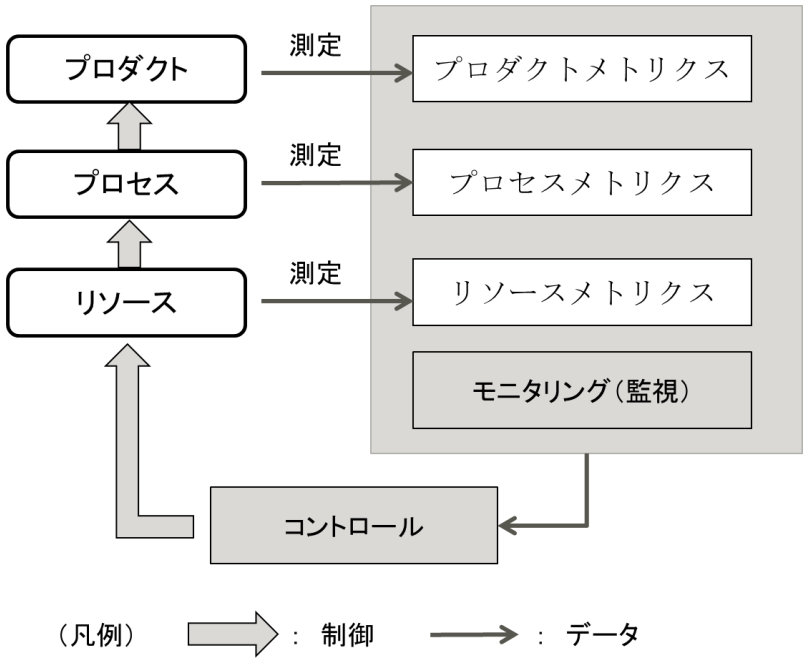


図 3-1　メトリクスの計測とプロジェクトマネジメント

（阿萬ほか（2011）の図1を元に、筆者がコントールの部分を追加、修正）

それぞれのメトリクスの概要を以下に示す。

(1) プロダクトメトリクス（Product Metrics）

ソフトウェア開発のプロダクト（成果物）について計測を行うメトリクス

(2) プロセスメトリクス（Process Metrics）

ソフトウェアライフサイクルの各フェーズやライフサイクル全体における活動について計測を行うメトリクス

(3) リソースメトリクス（Resource Mectics）

ソフトウェア開発で投入及び消費されるリソース（資源）について測定を行うメトリクス

表 3-1に、種類別のマトリクスの例を示す（阿萬ほか 2011）。

表 3-1 メトリクスの具体例

No.	メトリクスの種類	メトリクスの具体例
1	プロダクト	ソフトウェア規模（Function Point、ソース行数、画面数等）、複雑度、CK (Chidamber-Kemere) メトリクス、コードクローン数、欠陥数、欠陥密度
2	プロセス	工数、工期、生産性、仕様変更回数、ソースコードの変更回数、レビュー時間、レビュー密度、テストケース数、テスト密度、テストケース網羅率、欠陥除去率
3	リソース	投入可能工数、工期（スケジュール）、費用（予算）、開発要員数、経験年数

以下にそれぞれの分類別に代表的なメトリクスを取り上げ、概要を述べる。

3.2.2. プロダクトに関するメトリクス

プロダクトに関するメトリクスの利用目的について、阿萬ほか（2011）は、以下のように述べている。

(a) プロダクトに関するメトリクスは、対象成果物の属性を定量的に把握するためのものである。

(b) 得られた情報を解析することで品質の評価や予測に活用される。

(c) ソフトウェアを構成するモジュールに潜在する欠陥の予測や品質の評価を行うことによって、その結果をテスト設計やレビュー計画に役立てる。

上記の目的で活用される主なメトリクスを、Jones (2008)、Marciniak (994)、野中ほか（2008）、阿萬・山下（2010）等の研究や文献から抽出し概要を説明する。

3.2.2.1. ソフトウェア規模に関するメトリクスについて

ソフトウェアの規模は、開発工数の見積りやスケジュール策定などの基本となる重要なメトリクスであるが、ソフトウェアには物理的な量や大きさがないため、その規模を定量的に表す方法が模索され、様々な手法が提案されてきた。

Briand and Wieczorek (2002) は、ソフトウェア規模計測の方法を、「問題指向」(Problem-Oriented) と「ソリューション指向」 (Solution-Oriented) の2種類のタイプに分類している。問題指向のメトリクスは将来のソフトウェアが解決する問題、すなわち、システムの目標となる機能の大きさを表すものである。このメトリクスは、多くの場合、ソフトウェアの機能面の要素を計測することに焦点を当てているため、一般的に、ソフトウェアの「機能規模」と呼ばれる。実装方法などに依存しないことが特長である。他方のソリューション指向のメトリクスは、仕様書、設計書、ソフトウェアシステムの実装としてのソースコードなどのソフトウェア開発の成果物の大きさを表すものであり、その大きさは記述方法や実装方法、開発言語等に依存したものになる。

表 3-2に、Trendowicz and Jeffery (2014) によって、前者の機能に関するソフトウェア規模のめメトリクスとして分類されている手法の一覧を示す。

表 3-2　機能に関するソフトウェア規模のメトリクス (Trendowicz and Jeffery 2014)

（Trendowicz and Jeffery (2014) によるAppendix Table Aを元に筆者が解説を追加）

No.	手法	説明
1	Function Point (FP)	Albrecht (1979) によって提唱された、ソフトウェア機能の相対的な大きさを測定する手法。ソフトウェアによって生成されるデータのタイプに応じた数値の合計によって計測される。 複数のバリエーションの計測手法が提案され、ISO標準となっている手法として、下記の標準規格がある。 [1] IFPUG法: ISO-20926 (ISO 2009) [2] NESMA法: ISO-2450 (ISO 2005) [3] FSMA: ISO法-2981 (ISO 2008) [4] COSMIC法:　ISO-19761 (ISO 2003) [5] Mk II: ISO法-20968 (ISO 2002)
2	フィーチャーポイント	Jones (1996) によって提唱された。Function Pointを拡張したものであり、アルゴリズムの数に相対的複雑度を乗ずることによって求める。

3	ユースケースポイント	Karner (1993) によって提唱された。オブジェクト指向分析開発手法で用いられる表記法としてのUMLに含まれる「ユースケース仕様書」を入力情報元として、利用者としての「アクター」と利用場面に対応した「ユースケース」の「インスタンス数」（機能数）を数える。Function Point法と同様に複雑度に応じた重みを割り当て、重み付け総和を求める。ここで、「ユースケース」とは、システムに対する利用者の操作例を示したものである。
4	ストーリポイント	アジャイルソフトウェア開発プロジェクトのソフトウェア規模の計測を目的として、Cohn (2004, 2005) によって提唱された。ストーリポイントは、開発工数と複雑度が複合した直感的な相対的な大きさの単位である。 ストーリポイントを求める手法として、「プランニングポーカー」という手法がある。これは、ストーリポイントを表すための数字が書かれたカードを使って、チーム内の開発者が見積もったポイントを提示し合い、意見交換しながらストーリポイントを決める方法である。
5	オブジェクトポイント	ソフトウェアの大きさを表すための複数のオブジェクトポイントが提案されている。 [1] Predictive Object Points (POP): Minkiewicz (1997) によって提唱された、クラスの数とメソッドの重み付けによる計測を基本とするソフトウェア規模のメトリクスである。継承ツリーの平均的深さや、クラス当たりの子クラスの平均的クラス数によって調整される。 [2] Object Points (OP): Banker et al. (1992) によって提唱された。統合CASEツールにおける第 4 世代言語によって開発されるビジュアルウィジェット (widgets) の数を計測することによってソウェアの大きさを求める手法である。ウィジェットの具体例として、(1) 論理的なシステムコンポーネント、(2) 第四世代言語用部品、(3) ユーザインタフェース画面、(4) 帳票等がある。ウィジェットの数に対して、複雑性や再利用性を考慮した係数による調整が加えられ、全体のオブジェクトポイントが集計される。

6	Webオブジェクト	Reifer (2000) によって提案された。Webオブジェクト手法は、FP法（Function Point 法）の計上に利用される5つの要素に、Webアプリケーションに特有な4つの要素を追加してFP法を拡張したものである。追加された要素としては、マルチメディアファイル、Web構成要素、スクリプト及びリンクがある。Webオブジェクトは、ユーザ要件やWebページデザインの情報から計測される。計測手法は、FP法に類似しており、複雑度によって重み付けされた個々の要素数の合計によって規模が計上される。

表 3-2のNo.1のFunction Point法は、広く普及している手法の一つであるため、具体的な計測方法については後述する。

次に、表 3-3に、Trendowicz and Jeffery (2014) によって、「構造に関するソフトウェア規模のメトリクス」として分類されている代表的なメトリクスを示す。

表 3-3　構造に関するソフトウェア規模のメトリクス (Trendowicz and Jeffery 2014)

（Trendowicz and Jeffery (2014) によるAppendix Table Bを元に筆者が解説を追加）

No.	手法	概要
1	LOC (Lines of Code)	LOC (Lines of code) 法は、プログラムのソースコードのテキストの行数を計上することによってソフトウェア規模を計測するものである。LOCメトリクスには多くの種類があるが、それらの違いは、主として何をコードの行数として計上するかによるものである。Park (1992) は、異なるLOCメトリクスごとのチェックリストのサンプルを提供している。 以下はLOCの計測方法の例である。 [1] 物理的LOC (pLOC) ： コメントや空白を含む、「行末」の文字を有する全てのコード行数を計測する方法。 [2] 有効LOC (eLOC) ： コメント、空白や単独の角括弧や括弧を除く、行末の記号を有する全てのコード行数を計測する方法。 [3] 論理LOC (LOC) ： 命令行を形作るコード行数を計測する方法。命令行は、セミコロンなどの、各プログラミング言語に特有の終端記号で終了するものとする。

		[4] 拡張LOC (EnhLOC)： 改変されたソフトウェアの大きさを計測する方法。追加、削除または修正されたコード行数を別々に計測する。
2	Halsteadメトリクス (Halstead's Science)	Halstead (1977) によって提唱された。ソフトウェアコードの大きさと複雑度に関する複数のメトリクスを含む科学的方法。 [1] Halstead長： プログラム長（N）は、ソフトウェアのプロフラムコードの中のオペレータの総数（N1）とオペランドの総数（N2）の合計である。 [2] Halstead量： 数学的ビットで計測されるプログラムの情報を代表するプログラムの量（V）。Halstead長（N）とボキャブラリサイズ（n）の2を基底とするLogの掛算によって計上される。ここでボキャブラリサイズ（n）は、全ての単一の重複のないオペレータ（n1）と単一の重複のないオペランド（n2）の合計である。 Halstead量は、アルゴリズムの実装の大きさを表すものである。
3	プリミティブな計測手法	ソフトウェアの成果物の構造的要素を単純に計測する方法。このアプローチの例としては､ソフトウェアのソースコードのファイル数、要件定義書の中の要件数やUML仕様書の中のアクティビティ数等がある。

3.2.2.2. その他のプロダクトに関するメトリクスについて

ソフトウェア規模以外のメトリクスとしては、表 3-1に示す、複雑度、モジュール強度、モジュール結合度、テストケース数、テスト密度、テストケース網羅率、バグ数、欠陥密度等がある。これらのうちの具体的なメトリクスの概要について以下に示す。

(1) McCabeメトリクス（サイクロマティック数）

フローチャートまたはソースコードを対象として制御フローの複雑さを測定する方法である (McCabe 1976)。

(2) CK (Chidamber-Kemere) メトリクス

オブジェクト指向設計を対象としてその複雑さや継承木の深さ、結合度（coupling）凝集度（cohesion）といった複数の属性を測定して定量化する (Chidamber and Kemerer 1994)。

(3) コードクローン（複製されたソースコード）の個数、最大長、割合

保守性の良いソフトウェアを開発するためには、なるべく共通の機能を共通部品として開発することによって、機能追加を行う際の影響範囲を局所化することが望ましい。しかし、実際には、機能追加を行う場合には、類似したモジュールのソースコードを複製して別のモジュールを開発する方法を採用することもある。このような開発方法を繰り返すと、機能追加を行う場合に多くのモジュールの修正が必要になり、ソフトウェアの規模が増加することによって、保守性が悪化する。したがって、コードクローンの個数や割合を計測することによって、保守性を評価するための指標とすることができる。

(4) コメント文記述の密度、頻度

ソースコードの中に記述されたプログラムのロジックや変数についての説明文などのソースコード全体に占める割合や頻度を計測することによって、第三者がソースコードを理解しやすいか否かという観点から、ソースコードの保守性や品質の評価指標として利用する。

(5) 欠陥数、欠陥密度

ソフトウェアの品質を評価するためのメトリクスとして、レビューやテストで発見された欠陥の件数や、欠陥密度（単位規模当たりの欠陥数）が用いられる。

これらのソフトウェアのプロダクトに関するメトリクスは、作成されたソフトウェアのソースコードを元にして、品質や保守の容易性等の分析に利用されることが多く、プログラムの構造設計や、テスト工数の見積り、品質の評価などに活用可能である。ただし、プログラムが作成された後でないと測定できないため、設計工程でのプロジェクトマネジメントでは活用が難しい。本研究では、主として変動の把握や工数見積りのため、設計の段階から活用可能な機能に関するソフトウェア規模のメトリクスを活用する。

3.2.3. プロセスに関するメトリクス

プロセスに関するメトリクスは、ソフトウェアライフサイクルにおける活動を測定対象としたメトリクスである（阿萬ほか 2011）。活動の定量的な把握・分析によって、状況判断や評価、予測に役立てる。代表的な例としては、生産性の評価や進捗の評価が上げられる。プロダクトメトリクスによって成果物の規模を計測し、プロセスメトリクスによって時間や工数が算出される。これらから単位時間や単位工数当たりの規模として生産性が評価される。レビューやテストでのレビュー密度、欠陥指摘率などのメトリクスも、レビューの実施という活動の品質を評価、管理するために使われる。また、開発終了後のアンケートによって、失敗の要因を分析し、プロジェクトの失敗の未然防止に役立てるための研究も多数行われている。

(1) プロセスメトリクスの例

プロセスメトリクスの例として、以下のものが挙げられる。

(a) 工数、工期、進捗率

(b) 生産性

(c) 仕様変更回数、仕様変更率

(d) ソースコードの変更回数

(e) 統計的解析ツールの使用・不使用

(g) レビュー密度

(h) テスト密度、テスト網羅率

(i) 欠陥除去率

(2) プロセスに関するメトリクスとプロジェクトマネジメントの関係について

プロセスに関するメトリクスは、プロジェクトマネジメントにおけるコントロールの指標として利用するものが多い。

また、物理的に捉えることが難しいソフトウェア開発においては、品質をどのように定量的に把握し、管理するかも重要である。プロセスについての品質管理に関するメトリクスとして下記のものがある。本研究は、主にプロジェクトの変動として現れる進捗やコストの定量的な計測や制御を対象としているが、例えば、設計工程における品質の良否によって、製造工程やテスト工程の進捗やコストに影響が生ずるため、変動に対処するための原因分析や対策を行う際に、レビュー密度、テスト密度、テスト網羅率、欠陥除去率等のプロセスの品質に関するメトリクスや、欠陥数、欠陥密度等のプロダクトの品質が利用される。

これらの品質評価のためのメトリクスは、本研究で直接扱うものではないが、提案する変動のマネジメント手法を活用する際に、原因分析や対策検討の際に利用するものとなる。また、これらの品質に関する指標の元になるのは、ソフトウェアの規模であるため、各メトリクスを用いて正しい評価を行うためには、定期的に規模の変動を把握し、最新の規模を正確に把握することが重要となる。変動のマネジメントを対象とする本研究は、こうした、各種のメトリクスを正しく活用することによってプロジェクトマネジメントの成功率を高める上で、重要な意義を有するものである。

本研究では、プロジェクトの様々な要因による変動が、結果として表れる工数、工期（スケジュール）や生産性等を、変動に対するマネジメントのための主たるメトリクスとして活用することに焦点を当てる。その他のメトリクスは、変動が発生した際の原因究明の指標として活用する。

3.2.4. リソースに関するメトリクス

リソースに関するメトリクスとしては下記のものが挙げられる。

(a) 投入可能工数

(b) 工期（スケジュール）

(c) 費用（予算）

(d) 開発要員の生産性

(e) 経験年数

本研究の対象であるプロジェクトの変動のマネジメントにおいては、これらのメトリクスは、プロジェクトの計画と実績の差異を監視することによって変動を把握することや、コントールのための見積りや対策の効果を確認するために利用する。従来の多くの研究では、見積りや計画作成の精度向上を主たる目的として、各種のメトリクスの提案や改善が行われてきた。本研究は、プロジェクトの規模、進捗やコスト等の変動を実績データに基づいて把握し、生産性の見直しや計画の見直しのための工数や要員数の再見積り等に、これらのメトリクスを活用するための方法に焦点を当てたものである。

以上の定量的マネジメントのための各種メトリクスのうち、プロジェクトマネジメントの基礎的となるソフトウェア規模のメトリクスのうち、変動のマネジメントと関連性の強い機能規模を表す手法について、以下の節で説明する。

3.2.5. ファンクションポイント（Function Point）法

ここで、プロダクトに関するメトリクスの中でも特に重要なソフトウェアの機能の大きさを表すための代表的な手法であるファンクションポイント（Function Point）法の特長や計測方法の概要を説明する。主として、JIS X 0142:2010「ソフトウェア技術 - 機能規模測定 - IFPUG 機能規模測定手法（IFPUG4.1 版未調整ファンクションポイント）計測マニュアル」や情報処理推進機構の「ソフトウェア開発見積りガイドブック」(IPA 2008) を参考にしている。

(1) ファンクションポイント（FP法）法の位置付けと国際標準化

ファンクションポイント法（以下、FP法と略す）は、開発言語やアプリケーション方式等に依存しないソフトウェアの機能の量を計測することを目指して開発された手法である。データの入出力などの機能に着目し、機能数と重みによりソフトウェア規模を定量化する手法である。1979年にAllan J Albrechtが提唱した後、計測者による解釈の違い等が生じないように国際的な標準化が進められてきた。FP法の標準化団体として1986年に発足したIFPUG（International Function Point User Group）が定めた計測手法が国際標準としての「IFPUG法」である。世界で最も多く使用されているFP法の計測手法であり、日本ではJIS X 0142として標準化されている。このほか、ソフトウェアの分野の特性に合わせて、NESMA

法 (NESMA 2009) やCOSMICFFP法などの計測手法が提案されており、それらの手法についても国際標準化が進められてきた。

(2) IFPUG法の計測手法

IFPUG法の計測手法の概要を以下に説明する。以下の内容は、IFPUG発行の”Function Point Counting Practice Manual Release 4.3.1” (IFPUG 2010)（以下CPMと略す）に基づいたものである。

(a) 機能の種別について

FP法では、アプリケーションソフトウェア（以下、アプリケーションと略す）の機能をファイルに対する入出力データの観点で分類する。図 3-2に、IFPUG法による規模計測対象としてのアプリケーションのモデルを示す。図 3-2の中の、内部論理ファイル（ILF : Internal Logical File）、外部インタフェースファイル（EIF : External Interface File）、外部入力（EI : External Input）、外部出力（EO : External Output）、外部照会（EQ : External Inquiry）の5種について表 3-4に概要を説明する。以下では、これらの略称を用いる。

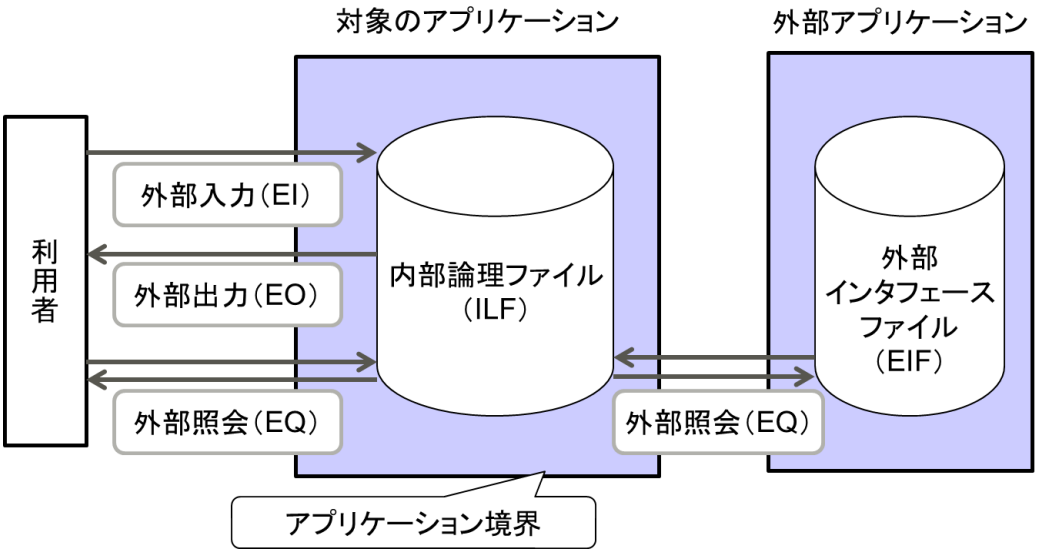


図 3-2 FP 法による計測対象としてのアプリケーションのモデル

表 3-4 機能種別に、各機能の概要を示す。

表 3-4　機能種別

分類	名称	略称（英語）	説明
データファンクション (Data Function)	内部論理ファイル	ILF (Internal Logical File)	アプリケーションで維持管理されるデータの集合
	外部インタフェースファイル	EIF (External Interface File)	外部のアプリケーションで維持管理され、参照のみ行うデータの集合
トランザクションファンクション (Transaction Function)	外部入力	EI (External Input)	ILFを更新するためのデータ入力機能
	外部出力	EO (External Output)	アプリケーション外部へのデータ出力機能
	外部照会	EQ (External Inquiry)	データ検索機能

データファンクション（Data Function）という、アプリケーションが使用するデータの集合として分類されるのが、内部論理ファイル（ILF）と外部論理ファイル（EIF）である。ここで、「ファイル」とはデータのまとまりの意味であり、物理的な電子ファイルの意味ではない。内部論理ファイル（ILF）は、データがアプリケーショ内で更新、追加、削除されるものであり、外部論理ファイル（EIF）は、参照のみが行われるのものである。次に、トランザクションファンクション（Transaction Function）という、データの処理機能として分類されるものが外部入力（EI）、外部出力（EO）、外部照会（EQ）である。トランザクションファンクションはデータ処理に関する機能であり、「アプリケーション境界」を超えてデータの入出力を行う機能である。図 3-2に示す「アプリケーション境界」とは、対象となるアプリケーションの範囲を示す境界である。外部入力（EI）は外部からのデータ入力によるデータの更新が目的であり、外部出力（EO）と外部照会（EQ）は利用者への情報提供を目的とするものである。また、外部出力（EO）は、アプリケーションの処理によって情報を利用者に提供するものであり、外部照会（EQ）は、単純なデータ検索によって情報提供をするものである。

(b) 重み付けについて

FP法では、個々の機能に対して重み付けを行うことによって個々の機能の量を数値化する。IFPUG法では、機能種別（ILF、EIF、EI、EO、EQの5種）と、機能毎の複雑さの組合せによって重み付けの数値を決定する。機能毎の複雑さは、単純、平均、複雑の3種類に分ける。これらの関係を図 3-3　機能種類別の複雑さに示す。複雑さについては単純をS、平均をM、複雑をCの記号で表している。

外部入力（EI）

参照ファイル種別数	データ項目数		
	1～4	5～15	16～
＜2	S	S	M
2	S	M	C
＞2	M	C	C

外部出力（EO）、外部照会（EQ）

参照ファイル種別数	データ項目数		
	1～4	5～15	16～
＜2	S	S	M
2	S	M	C
＞2	M	C	C

外部照会（EQ）の入力側

参照ファイル種別数	データ項目数		
	1～4	5～15	16～
＜2	S	S	M
2	S	M	C
＞2	M	C	C

外部照会（EQ）の出力側

参照ファイル種別数	データ項目数		
	1～4	5～15	16～
＜2	S	S	M
2	S	M	C
＞2	M	C	C

外部インターフェースファイル（EIF）

参照ファイル種別数	データ項目数		
	1～19	20～50	51～
＜2	S	S	M
2	S	M	C
＞2	M	C	C

（凡例）

S　：単純

M　：平均

C　：複雑

図 3-3　機能種類別の複雑さ

複雑さは、該当の機能のデータ項目やアクセスするファイル数を元に決定する。データファンクションについてはレコード種類数（RET：Record Element Type）とデータ項目数（DET：Data Element Type）の2種類の数値を計測する。トランザクションファンクションについては関連ファイル数（FTR：File Type Referenced）とデータ項目数の2種類の数値を計測する。これらのデータファンクションとトランザクションファンクションのデータ項目数のマトリクスを作成し、個々の機能の重みを合計することにより、アプリケーション全体のFP値が算出される。この方法によって算出された数値を「未調整ファンクションポイント」（Unadjusted Function Point）と呼ぶ。

IFPUG法では、この未調整ファンクションポイントの値に対して調整要因等を用いて調整済ファンクションポイントを計算する方法が示されているが、この調整要因以下の手順はオプションとされている。機能に関する規模を表す上では、未調整ファンクションポイントをもってFP法の計測値として用いることが多いため、調整済ファンクションポイントの説明は省略する。

(d) IFPUG法によるFP計測のための条件

IFPUG法によってFP計測を行うためには、下記の情報を得られる必要があり、データに関する設計が完了している必要がある。

・内部論理ファイルの情報（マスタファイルやデータベース）や外部インタフェースファイルの情報

・5つの機能（ILF、EIF、EI、EO、EQ）
・各機能の複雑度
・データファンクションのレコード種類数とデータ項目数
・トランザクションファンクションにおいてアクセスするデータファンクションの数と入出力データ項目数

(3) NESMA法

IFPUG法では、前述のように、機能別に各種のデータに関する情報が必要となるため、詳細なシステム設計が確定した段階にならないと正確な計測が行えない。そこで、開発の初期の段階でFPを計測できるようにすることを目的として、NESMA（Netherlands Software Metrics Association）によって、下記のFP概算法とFP試算法 (NESMA 2005) が提案されている。

(a) FP概算法（The Estimated Function Point Count）

FP概算法は、基本的な規模計測の考え方はIFPUG法と同じであるが、データ設計が完了していない段階でも計測できるようにするため、機能ごとの重み付けにおける複雑さの判定を以下のように簡略化する。
・データファンクションの複雑さを全て「単純」とする
・トランザクションファンクションの複雑さを全て「平均」とする

FP概算法は、複雑さの計測だけを簡略化しているものであるため、一つの業務システムの中で機能による複雑さの偏りが少ない場合には、IFPUG法による計測値との乖離が少なくなると考えられる。

(b) FP試算法（The Indicative Function Point Count）

FPs試算法では、IFPUG法の計測法におけるデータファンクションの個数のみを計測し、その結果を以下の計算式に当てはめてFP値を算出する。

FP試算値 ＝ 35 × ILFの個数 ＋ 15 × EIFの個数

ここで、35及び15という係数は、ILFとEIFの平均的なトランザクションファンクションの数を仮定して機能全体のFP値の合計を算出するためのものである。具体的には、内部論理ファイル（ILF）には3つの外部入力（EI）として、ILFに対する追加、変更、削除と、2つのEO、1つのEQが含まれると仮定する。外部インタフェースファイル（EIF）には1つの外部出力（EO）と1つの外部照会（EQ）が含まれると仮定する。このような仮定に基づいた計測法であるため、IFPUG法による計測値との乖離が大きくなることがある。

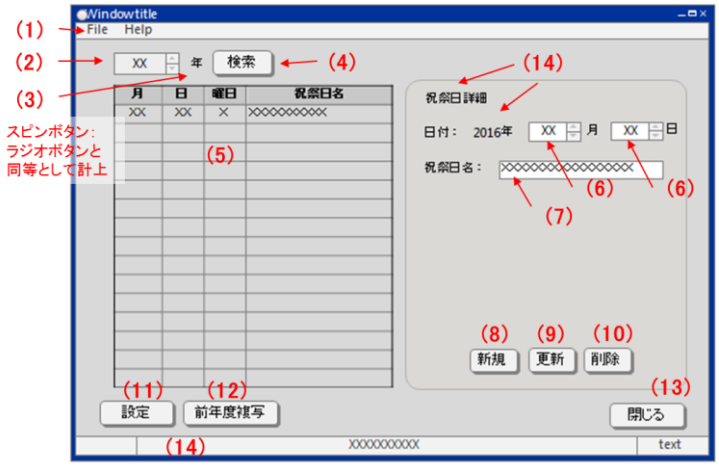
(c) 設計・開発の工程によるFP計測法の選択及び組合せ

各種のFP計測法の中から、設計や開発の工程に応じて適した手法を選択して組み合わせて適用する方法がある。例えば、データ項目が明らかになった時点でFP試算法によりデータファンクションの数に係数を掛けて規模を見積る。画面や帳票などの機能の設計が進んだ階で、ファイル数の情報が不明確な場合には、FP概算法により複雑さの判定を簡略化することによって概算FPを求めることできる。データに関する設計によって、データ項目や各機能別のアクセス対象テーブル数などが明確になった時点でIFPUG法による詳細なFPを計算することが可能となる。このように、設計工程の進捗に応じて適切な手法を組み合わせる方法がある。ただし、工程によって使い分けることによって、計測手法によって計測値に乖離が発生する可能性がある点については、規模の変動状況を把握する際に注意が必要となる。

3.2.6. FS (Function Scale) 法

ソフトウェアの機能を定量化するための手法として、前述の FP 法のほかに、ファンクションスケール (Function Scale) 法（以下、FS 法と略す）がある。

画面レイアウト



ＦＳ集計表

No.	ラベル名	コントロール	FS
1	ファイル，ヘルプ	プルダウン	30*2
2	検索年	ディスレイフィールド	5
3	年増減	スピンボタン	10
4	検索	ボタン(検索)	100
5	結果表示	リストボックス	30
6	月，日	プルダウン(相関チェック有)	30*2 +10*2
7	祝祭日名	エディットフィールド	15
8	新規	ボタン(登録)	130
9	更新	ボタン(更新)	150
10	削除	ボタン(削除)	115
11	確定	ボタン(更新)	150
12	複写	ボタン(検索)	100
13	閉じる	ボタン	5
14	祝祭日詳細，2003，メッセージ領域	ディスレイフィールド	5*3
計			965

図 3-4　Web システムの画面レイアウトの例と FS 集計表

（富士通（2013）の「ファンクションスケール法　測定マニュアル」より）

FS 法は、ソフトウェアの外部設計情報を元にして機能規模を定量化する手法であり、簡易的な FP 法の一種として位置付けられる（富士通 2009）。FS 法では、システムの画面や帳票等の単位で機能規模を算出し、規模の単位は“FS”を使う。FS 法は主として Web システムの規模の把握に適した手法であるが、他のシステム形態でも適用可能である。

図 3-4 に Web システムの画面レイアウトの例を示す。図 3-4 の右側の表は、左の画面の FS 値の集計表である。画面上の入力フィールドやボタンに番号を付与しており、それぞれの数字に対応した画面の構成要素ごとの FS 値を表に整理したものである。画面の構成要素であるディスプレイフィールド、エディットフィールド、リストボックスなどのコントロールの種類別に基準値（ポイント数）（単位：FS）が決まっている。ボタンについては、データベースアクセスの処理内容を考慮した定数を設定しており、検索、更新、登録などの機能によって基準値が異なる。

このように、画面の構成要素（コントロール）別の基準値に各構成要素の個数を掛けて集計することで画面単位の FS 値が求められる。帳票やバッチ形式のアプリケーションの規模については、構成要素としてデータ項目数を使うなど、構成要素の種類や定数が異なるが、要素ごとの定数を積み上げるという考え方は同じである。

FS計測（オンライン）ワークシート　プロジェクト名　作成日　作成者　(5.3版用)

SEQ	画面名	画面ID	エディットフィールド	相関関係チェック数	ディスプレイフィールド	ラジオボタン（グループ）	相関関係チェック数	チェックボックス	相関関係チェック数	リストボックス	相関関係チェック数	スプレッド	タブ	ボタン 検索	登録	更新	削除	印刷	遷移	表示変更	初期表示	オン詳細 FS
1																						0
2																						0
3																						0
4																						0
5																						0
6																						0
7																						0
8																						0
9																						0
10																						0
11																						0
12																						0
13																						0
14																						0
15																						0
16																						0
17																						0
18																						0
19																						0
20																						0
画面数 0	合計		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FS分布データ

FS値	~200	201~400	401~600	601~800	801~1000	1001~1200	1201~1400	1401~1600	1601~1800	1801~2000	2001~2200	2201~2400	2401~2600	2601~2800	2801~3000	3001~3200	3201~3400	3401~3600	3601~3800	3801~
詳細画面数	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図 3-5　Function Scale 法による規模計測用ワークシート

図 3-5 に、オンラインアプリケーション用の FS 計測用ワークシートの一例を示す。画面ごとのコントロールの数を記入することにより、画面単位やサブシステム全体の規模等が自動集計される。

次に、図 3-6 に、画面のレイアウトの一例と FS の計測値を示す。

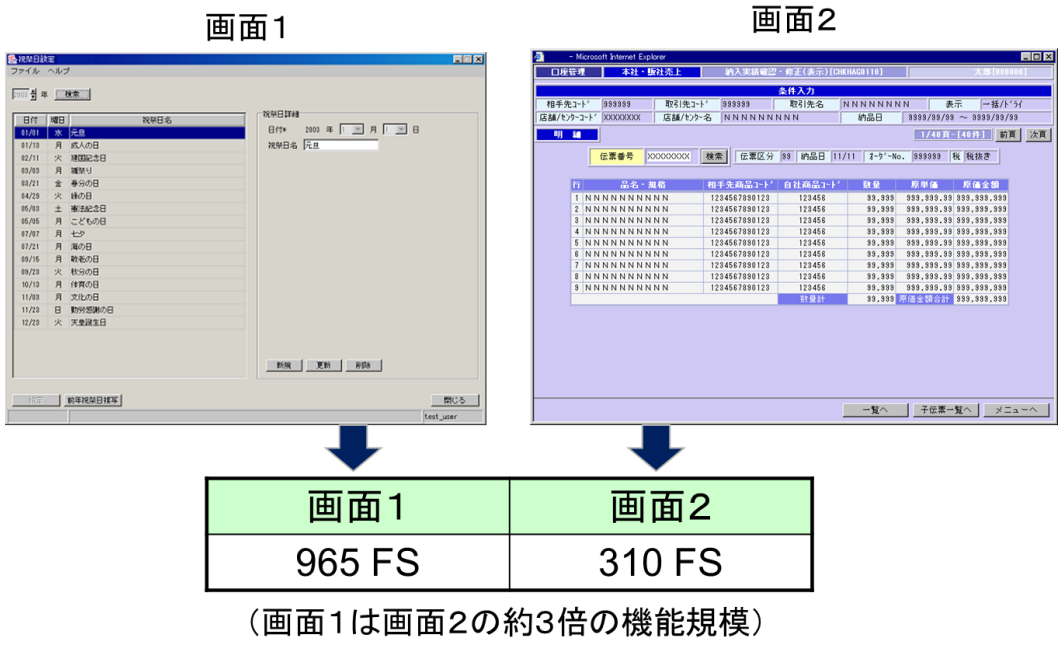


図 3-6　FS の計測事例の比較

（富士通（2013）の「ファンクションスケール法　測定マニュアル」より）

図 3-6 の 2 つの画面を比較すると、見かけ上は、画面１よりも画面２のほうが複雑な画面に見えるが、FS 法の計測値を比較すると、画面１のほうが FS 値は大きい。これは、画面１の方が、「更新」や「検索」などの、基準値の大きいボタンの数が多いことによる。このように、画面の見た目などに左右されずに、誰が計測しても同じ FS 値が得られる。

以上のように、FS 法では画面などの構成要素を元に機械的に規模を算出するため、測定者に依存せずにぶれのない機能規模を算出することができる。また、測定に要する時間に関しては、ある程度習熟すれば一画面当たり数分で算出できる。また、標準の「画面項目定義書」を使用して画面の外部仕様を設計することにより、FS 値を自動的に計算するための計測ツールを作成することも容易であり、規模計測の自動化が可能となる。

FS 法の特長を整理すると以下の 3 点に要約される。

(1) 画面や帳票に関する外部設計情報から画面や帳票の単位で規模を算出でき、利用者やプロジェクトオーナー等にとって理解しやすい

(2) 人に依存せずにぶれのない規模を算出できる

(3) 比較的軽易に測定できる

本研究における規模変動の可視化手法においては、FS法を採用した。この理由は、上記3つの特長により、外部設計情報から、ぶれのない機能規模を少ない負担で繰り返し測定できるため、規模の変動状況を継続的に把握する上でFS法が適していると考えたためである。ただし、提案する規模変動の可視化手法は、規模計測手法を特定のものに限定するものではなく、同様の特長を有する規模計測手法であれば、他の計測手法の利用も可能である。

3.2.7. ソフトウェアの機能規模とコスト見積りに関する課題と本研究での扱い

Jones (2008) は、「ソフトウェアの機能と要求は無形であり、複雑さの増大は生産性で計られるべきかファンクションポイント値の大きさで計られるべきかの問題となる。ソフトウェアの複雑さをどう扱えば良いかは、検討の余地がある。複雑度と構築の難しさは生産性及びファンクションポイント当たりのコストの調整によって扱うほうが良いように見える。」と述べている。これは、ソフトウェア規模のメトリクスとして比較的広く使われているファンクションポイント法（FP法）においても、ソフトウェアの様々な機能を全て定量的に表せるわけではないことを示している。例えば、FP法はデータやトランザクションの数を元に数値を算出しているため、ソフトウェアの状態遷移やユーザインタフェースなどの、これらの観点には数値として計上されない要素もある。FP法の主流になっているIFPUG法では、重み付けを、データ項目数やファイル数によって決まる3段階の「複雑度」として表し、ポイント数に計上している。しかし、FP法で定義されている「複雑度」には、上記のソフトウェアの動的機能やユーザインタフェースなどの複雑度は含まれない。実際には、開発に利用するソフトウェアの開発言語や、設計方法、システム全体のアーキテクチャなどによって、構築の難しさが変わるため、最終的に開発コストを見積もる場合には、これらの複雑性や構築の難しさを考慮する必要がある。

FP値を元に開発工数を見積もる場合には、

開発工数（人月） ＝ 生産性（人月／FP値） × FP値・・・①

という式によって見積もることが一般的である。ここで、生産性は、単位規模（FP値）当たりのソフトウェアの開発に必要な工数（人月）である。

次に、開発コストを、人件費を元に見積もる場合は、

コスト（円） = 単価（円／人月）×開発工数（人月）・・・②

という式で見積もる。ここで、単価は、１人月当たりの人件費を表す。

あるいは、FP値から直接コストを見積もる場合は

コスト（円） ＝ コスト単価（円／FP値）× FP値 ・・・③

という式で算出することになる。

この際に、FP値に含まれない複雑性や構築の難しさが増加した場合に、FP値を増加させるのか、式①における生産性が低くなることで表すか、式③におけるコストの単価の増加

で表すのかという問題になる。Jones (2008) は、「FP値に含めるのではなく、式②の生産性や式③のコスト単価に含めるのが妥当であろう」と述べている。

本研究では、ソフトウェアの複雑性などについては、ソフトウェア規模から工数を算出する際の生産性の補正に含める方法を採用した。生産性には、ソフトウェアの複雑度やアーキテクチャだけでなく、担当者のスキルや、開発作業の反復回数による習熟度など、様々な要素が影響する。従来の研究は、ソフトウェアの規模自体の精度の向上や、規模から工数を見積もる場合の工数見積り手法の精度向上に重点が置かれてきた。しかし、ソフトウェア開発においては、ソフトウェア自体の複雑性や製造の柔軟性等に加えて、人が関わる作業の生産性の影響などが大きく、見積りの条件を精密に設定しようとすると、データ収集の負荷が高く、実績データを蓄積しても、条件が異なる他のプロジェクトへの適用が難しい。

そこで、本研究では、見積りや当初の計画作成の際には、過去の類似プロジェクトの統計値等から当初の生産性を設定し、工数を見積もった上で、WBS別の実績データを継続して正確に把握し、実績としての生産性を用いて再見積りを行うことによってプロジェクトマネジメントの精度を向上させるという立場をとる。このためには、工数実績や進捗を機能、人別の詳細なWBSの単位で把握するための方法が重要となる。本研究では、工数の見積りやスケジュールに大きく影響するソフトウェア規模の変動を把握し、実績工数を元に生産性や開発期間の影響を考慮して工数の再見積りを行い、早い段階で計画や体制の変更を行うことによってプロジェクトを適切にコントロールする方法を提案する。

3.3. 工数見積りに関する先行研究

3.3.1. 工数見積り手法

工数見積りの方法としては、以下の方法がある。

(1) 類推法

過去の類似プロジェクトの実績を基にして見積る方法

(2) 積み上げ法

プロジェクトの成果物の構成要素を洗い出し、それぞれに必要な工数などを見積って積み上げる方法

(3) パラメトリック法

工数などを目的変数として、説明変数に規模や要因などを設定し数学的な関数として表す方法である。代表的な手法として、COCOMO II (Boehm et al. 2000) やCoBRA法 (Trendowicz 2013) がある。

表 3-5に、各見積り手法の比較を示す（真野・誉田 1993 ）。

具体的な工数見積りの手法については、様々な研究が行われている。伝統的な手法の多くの方法は、エキスパートやプロジェクトマネジャーの専門能力を工数見積りに活用する方法や見積りのためのガイドを利用する方法が多い。Kocaguneli et al. (2012) は、エキスパートの能力とガイドやチェックリストを組み合わせて利用する方法を提示している。また、エキスパートグループの意見を集めて合意形成を行うデルファイ法を適用した方法が提案されている。解決すべき課題の記述を元に個々のエキスパートが匿名で見積りを行い、その結果が合意されるまで繰り返される。Boehm and Farqubar (1981) によって、エキスパート間の情報交換を強化した拡張デルファイ法が提案されている。

表 3-5　見積り手法の比較

（真野・誉田「見積りの方法」より引用し、IPA (2005)の情報を一部追記）

	種類	長所	短所	前提条件
1	類推法	・初期見積りに適している	・参考とする過去のプロジェクトの特徴や制約などが明らかでないと適用が困難 ・客観性に欠ける	・過去のプロジェクトの規模や工数等の実績データベースと、プロジェクトのプロフィールなどの情報の蓄積が必要
2	積み上げ法	・特性が類似しているシステムを繰り返し開発している組織においては精度が高い	・プロジェクトのWBSや成果物等の構成要素を洗い出す際の網羅度と厳密度が見積り精度に影響する ・不確実な事項が多い初期段階では見積りの精度が低くなる ・客観性に欠ける	・過去の実績に基づいて構成要素を分類、整理しておく必要がある ・それぞれに対して規模や工数などの実績データが必要
3	パラメトリック法	・再現性があり客観的 ・前提が明確 ・シミュレーションが可能	・関係式の中に不確定な変数があると見積りの誤差が大きくなる	・過去の実績データベースに基づいて関係式を定式化し、精度向上のための検証を行うことが必要

定式化されたソフトウェア開発の工数見積りモデルでは、見積りプロセスにおいて様々なツールや手法が活用される (Jorgensen et al. 2009)。初期の見積りモデルの典型的なものは、回帰分析手法に基づいたものが多い (Yucalar et al. 2016, Huang.et al. 2008) 。Putnam (1978)

のSoftware Lifecycle Management (SLIM) や、Boehm et al. (2000) のConstructive Cost Model (COCOMO II) は著名なソフトウェア工数見積りモデルである。そのほかに、決定木による見積り手法 (Kocaguneli et al. 2012, Liu 2008)、ケースベースで分析や類推を行う見積り手法 (Liu 2018, Idri 2015)、ニューラルネットワークを適用した見積り手法 (Kumar et al. 2008, Huang and Chiu 2009)、サポートベクターマシンを適用した見積り手法 (Oliveira 2016, Corazza et al. 2011)、ファジーロジックの観点を取り入れた見積り手法 (Mittal 2010, Muzaffar and Ahmed 2010)、進化的アルゴリズムを応用した見積り手法 (Oliveira 2015, Minku and Yao 2013) 等の研究がある。

エキスパートによる分析は、多くのソフトウェア企業において良く使われてきた。他方で、定式化されていない見積りモデルの利用は減少している (Jrgensen 2004)。Jorgensen and Boehm (2009) は、最適な工数見積りを行うためには、エキスパートによる分析と定式化されたモデルベースの見積り手法を組み合わせて利用することを提案している。これは複数の見積り手法による見積りを行ない、その組み合わせの過程で見積りの精度を改善する方法である。

Mendes (2014) は、様々な分野の有識者の知識を元に、Baysian Networkを基礎とした見積りモデルを構築することによって、それぞれの分野に適した精度の高い見積りが可能となることを示している。Trendowicz and Jeffery (2014) は、表 3-5に分類される様々な具体的な工数見積り手法について、適用する工程やプロジェクトの特性や環境に応じて最適な工数見積り手法を選定するための観点を網羅的に提示している。

Berlin et al (2009) は、ITプロジェクトにおける各種の工数見積り手法を比較しており、その結論として、早い工程で精度の高い見積りを行うためには、工数見積りに影響するプロジェクトの属性をなるべく正確に収集することが重要であると述べている。しかし、プロジェクトの属性は、後工程になると変化する場合が多い。特に開発プロジェクトにおいては、工程の進展に伴って曖昧な要件が明確になり制約が明らかになることで、仕様の追加や複雑性の増加が発生し、ソフトウェア規模を含む各種のプロジェクト属性は変動する。したがって、プロジェクトの早い段階の工数見積りでは、それらの属性自体が推定値であり、過去のプロジェクトのデータ等を活用した機械学習などによって、見積りの精度を高める必要があると述べている。

Kirmani (2017) は、近年のWebシステムの開発の工数を早い段階で精度良く見積もるために適切な手法を選択するための研究を行った。Kirmani (2017) は、工数見積りにおいて重要なことを3点挙げている。一点目は、できるだけ概算見積りと実績の差異が少なくなるような見積りモデルを選定すること、二点目は、見積りモデルが概算見積りと実績の差異の監視や開発プロセスのコントールに役立つものであること、三点目は、概算見積りと実績との差異が発生した時の分析のため、個人レベルでの詳細な工数見積りの根拠を透明にしておくことと述べている。本研究における変動の可視化とコントロールの手法は、Kirmaniの提案と同様の考え方を採用している。すなわち、変動の早期把握や変動の原因分析等を効

率良く行うための各種のデータを個人や詳細な作業の単位で蓄積し、各管理要素間の関係性を明確にしておくという考え方であり、さらに、各原因に応じた対応策を過去の経験からパターン化して形式知化することで経験の少ないプロジェクトマネジャーが過去の知識を活用可能とするものである。

3.3.2. 再見積りの手法

前節の Berlin et al. (2009)の研究等において言及されているように、プロジェクトの工程の進展に伴い、要件や制約が明確となることから、工数見積りに影響のある各種の条件やソフトウェア規模が変動する。Jodpimai (2018) は、プロジェクトの工程が進むに従って計画時の見積りと実績の間に変動が発生することへの対策として再見積りを行うことが重要であるとして、再見積り手法の精度について研究している。再見積りの考え方は、本研究の対象としているプロジェクトの変動に対するコントロールを行う際に重要となる考え方である。再見積りに関する研究は限られているが、下記の研究がある。

例えば、MacDonell (2003) は、前工程の工数を入力情報として、回帰分析によって次工程の工数を見積もる手法を提案している。Ferrucci et al. (2010) は、前工程の工数実績と合わせて、Function Point を組み合わせて再見積りを行うことで次工程移行の再見積りの精度を高められることを検証した。Azzeh et al. (2010) は、相関ルールやファジー集合の概念を利用するで見積りモデルを構築し、工数見積り精度を向上させることができたことを示した。Emran et al. (2010) は、ソフトウェアのリリース計画の再作成において、機能性と品質向上のトレードオフを行う上での再見積りの有効性をシミュレーション等で検証している。

3.3.3. 工数見積りや変動マネジメントのためのソフトウェア規模計測手法

ソフトウェア開発の計画策定や変動の監視、コントロールの基礎となる重要な工数見積りの観点で、ソフトウェア規模見積り手法の比較、検討を行った先行研究を取り上げる。

表 3-2及び表 3-3に示されるソフトウェア規模のメトリクスには、手法によって利点や欠点が異なるため、利用目的に適した手法を選択して適用する必要がある。Trendowicz and Jeffery (2014) は、工数見積りに適した規模見積り手法を選定するための基準について、ソフトウェア開発の工数見積りの観点で評価している。この際、Sheetz et al. (2009) によって定義された基準を基にしている。

構造に関する規模のメトリクスの一つであるLOC (Lines of Code) は、見積りにあたって経験者の知識が必要となることや、同じ機能のソフトウェアであっても開発言語の種類やソースコードの記述方法、プログラム構造の共通化の程度によって行数が異なることなどの欠点が指摘されている。それにもかかわらず、計測が比較的容易であるということから多くのプロジェクトで利用されている。適用にあたってはその欠点について注意深く考慮しながら利用し、そのメリットを生かせる場面で利用すれば、計測ツールなどを利用する

ことで容易に行数を集計できるという利点を生かした利用が可能となる。

他方、機能に関する規模のメトリクスとして普及しつつあるFP法は、LOCに比べて多くの場面において比較的良い結果を得ることができるとされている。しかし、弱点も存在するため、特定の状況においては事前に検討した上で利用する必要がある。そこで、表 3-6に、ソフトウェア開発の工数見積りにおけるLOCとFP法及びFS (Function Scale) 法の主な特徴の比較を示す。表 3-6は、Trendowicz and Jeffery (2014)のAppendix Table Cに対して、3.3節3.2.6のFS法の特徴を追加したものである。

表 3-6　LOC と FP 法及び FS 法の強みと弱みの比較

（Trendowicz and Jeffery (2014)によるAppendix Table Cを元に筆者がFS法の特徴を追加)

No.	観点	特徴	LOC	FP法	FS法
1	自動化 (Automation)	(1) LOC計測については、ソースコードを解析することにより容易に自動化できる。 (2) FP計測については、ソフトウェアの要件定義や仕様が自然言語を用いて非定形フォーマットで記述されている場合は自動化することは難しい。 (3) FS法は、設計書として良く使われる画面のデータ項目定義やイベントの定義を表形式の定型フォーマットを元に計測を自動化することも可能である。	＋＋	－－	＋
2	計算容易性 (Calculation ease)	(1) LOCは、完成したソフトウェアのコード行数を集計するだけで容易に計測可能である。この際、実行可能な行とコメントを分けて集計する機能を有する各種の計測ツールがOSS (Open Source Software)や製品として提供されている。 (2) FP法は、データとレコードの要素タイプの個数によって決まる相対的複雑度によって重み付けされたデータタイプの数を集計する必要があるため、計算が複雑である。 (3) FS法は、システムの画面レイアウト定義とコンポーネントの種類別の定数を合計するだけであり、比較的容易に計算できる。	＋	－	－ / ＋

3	具体性 (Objectivity) 及び計測の安定性（筆者が追加）	(1) LOCは、物理的なソフトウェアの成果物の構造的な属性を直接計測するため、FP法よりも具体的である。計測ルールが定義されれば、繰り返し計測することができる。 (2) FP法は、仕様書などに基づいてソフトウェアの抽象的な属性（機能）を計測する。機能的な要素を特定するためには、プロジェクトに特有の成果物の主観的な解釈を必要とする。このため、FP法の出力は、計測者によって、設計書などに記述された情報の解釈に違いが発生することがあり、いつ誰が計測しても繰り返し同じ結果を得られるかという観点については不十分である。 (3) FS法は画面レイアウトなどの利用者が使用するソフトウェアの目に見える機能との対応が明確であり、利用者の視点での具体性が高い。	＋＋	－	＋＋
4	データ入手性 (Data availability)	(1) LOC測定のためのデータは、ソフトウェアのテキストコードを解析することで容易に入手できる。 (2) FP法の計測のためのデータとしては、自然言語による非定形のソフトウェア機能仕様書等を必要とする。 (3) FS法で測定する対象となるデータは、画面レイアウト設計書やデータ項目定義書などのシステム開発で作成される設計書から抽出可能である	+	－	－
5	コンテキスト独立性 (Context independence)	(1) LOCの値は、ソフトウェアの開発者や適用技術への依存性が強い。例えば、同一の機能のソースコードであっても、プログラマのスキルによってコードの行数が異なり、プログラミング言語によってもLOCの数値は異なる。 (2) FP法は、ソフトウェアの機能を測定する	－－	＋＋	＋＋

		方法であり、コンテキストやコードの実装方法等には依存しない。 (3) FS法はFP法と同様にソフトウェアの機能を測定する方法であるため、コンテキストや実装方法には依存しない。			
4	ライフサイクル (Life cycle)	(1) FP法は、ソフトウェアの機能についての情報入手が可能となる早い工程から適用できる。機能を記述するドキュメントの様式には依存しない。FP法では、ソフトウェアの機能的なコンポーネントを定義するための様々な異なる成果物を利用する。 (2) FS法は、FP法と同様に、画面レイアウトや画面遷移図、データ項目定義等の設計書をもとにソフトウェアの機能を設計するライフサイクルの早い段階から、規模の計測が可能であるため、仕様の確定度合いに応じた規模の増減等の変動の把握に適している。 (3) LOCは、ソフトウェアコードのみに対して適用可能である。したがって、コーディング工程の終了より前に計測することはできない。このことにより、LOCを見積りの目的で利用する上での利便性は制限される。	－	＋＋	＋＋
5	問題の特定 (Problem Identification)	(1) FP法は、ソフトウェアの要求仕様のレビューや、不完全または不整合なプロジェクトスコープの特定のために利用できる。さらに、スコープクリープのモニタリングや、開発工程を通じた機能のトレースにも利用可能である。 (2) LOCは、ソースコードを記述した後でないと計測できないため、要件定義や仕様設計の段階での問題を特定することには適用できない。	－	＋＋	＋＋

6	改善の支援 (Improvement Support)	(1) FP法は様々な改善の支援が可能である。例えば、ライフサクルに渡って、異なる工程や異なる成果物の間で機能規模のトレースができるため、潜在的なスコープのクリープや機能的なトレーサビリティ上の問題の発見が可能となる。 さらに、コンテキストに対する独立性は、生産性や欠陥について、異なるコンテキストのプロジェクト間の比較を可能とする。 (2) FS法もFP法と同様の各種の改善に活用可能である。	－－	＋＋	＋＋
7	敏感性 (Sensitivity)	(1) FP法、FS法、LOCともに、計測値がソフトウェア要件に応じて変化する。 (2) LOCは、ソフトウェアの技術の変化にも影響を受けるが、FP法やFS法は実装の技術には影響されない。	－	－	－
8	直感性 (Intuitiveness)	(1) LOCは、ソースコードというソフトウェアの成果物の有形の特性を表すものであり、異なるプロジェクトのステークホルダにとって比較的直観的に把握可能である。ただしそれは、あくまでも結果としてのソースコードの物理的な量である。 (2) FP法は、ソフトウェアの抽象的な特性を表現するものであるため、開発者よりもソフトウェア開発のマネジャーや顧客にとって直観的に把握可能である。 (3) FS法は、画面レイアウト上のコンポーネントの種類や工数、データ項目数と規模の数値が対応しているため、いずれのステークホルダにとっても直感的に把握可能である。	＋	－	＋＋
9	理解容易性 (Understandability)	(1) LOCは、その意味合いや構造は比較的理解しやすい。コード行数や計測プロセスは、	+	－	＋＋

		シンプルである。ただし、仕様とコード行数の因果関係は、対象となるドメインのシステム開発経験のあるプログラマ以外のステークホルダにとっては理解しにくい。 (2) FP法については、規模のメトリクスの定義や計測手順が比較的複雑で、その意味や構造を理解するために学習のための時間を要する。 (3) FS法は、画面上のコンポーネント毎の数値が事前に示されているため、いずれのステークホルダにとっても理解が容易である。			
10	検証性 (Validity)	(1) FP法は、理論的な検証性には制約がある。例えば、特定のPF法によって、規模は理論的に加算可能ではない。また、FP法は、計測者によって仕様書の理解や解析の差に基づく計測値の差異が発生するため、実用的な検証性に制約がある。個人別の計測値には、最大で30%程度の差異が発生する。 (2) FS法は、画面上のコンポーネントの種類毎に定義された数値を合計するだけで計測可能であるため、計測者による計測値の差異は発生しない。規模の数値は単純に加算できるため、コンポーネントの種類をルール通りに判断すれば、誰が何度計測しても同じ数値になる。 (3) 他方、LOCは論理的には有効であっても、コーディング工程より以前の段階では実用面での検証性は低い。コーディングより前の工程で、LOCを見積る場合は、ほとんど見積り担当者に依存する。このため、見積り担当者によってソフトウェア規模の大きい乖離が発生し、工数見積りの目的での有用性は限られたものになる。	＋	＋＋	＋＋＋

（凡例）LOC、FP法、FS法の蘭の「＋」と「－」は各手法の強み、弱みの度合いを表しており、「＋」の数は強みの度合い、「－」の数は弱みの度合いを表す。

表 3-6に示すように、FP法とFS (Function Scale) 法は、ソフトウェアの機能に関する規模を表す手法であり、特徴が類似しており、工程の早い段階から要件や仕様に対応した規模の変動を把握する上で適している。計算容易性、具体性、直観性、理解容易性及び検証性については、FP法よりもFS法のほうが強みの度合いが大きいと言える。

3.4. 計画策定手法

計画に関する手法として、以下の方法がある。

(1) PERT

PERT (Program Evaluation and Review Technique) は、工程計画管理手法の 1 つで、プロジェクト全体を構成する各作業の相互依存関係を、PERT 図（アローダイアグラムとも呼ばれる）というネットワーク図で表すことによって、各作業の日程計画の作成、全体の所要時間の算出や、作業期間の短縮の検討等に活用する手法である。

ブーズ・アレン・ハミルトンが 1958 年、アメリカ国防総省 US Navy Special Projects Office からの委託の一環で、ポラリス潜水艦発射弾道ミサイルプロジェクトの一部として発明したものである。

1962 年には米国防総省と NASA（米国航空宇宙局）が PERT に原価管理の要素を加えた「PERT/Cost」を発表した。これが、後に EVM (Earned Value Management) へ発展する。さらに信頼性の要素を追加した「PERT/Reliability」や、人材管理に応用した「PERT/Manpower」などが登場した。

(2) CPM

PERTとほぼ同時期に独立して、デュポン社によってCPM (Critical Path Method) が開発された。PERTが日程だけを考えた計画法であるのに対して、CPMは最低限のコストの計画作成を狙った計画法である。ネットワーク図を用いてスケジューリングを行うことから、PERTと合わせてPERT/CPMと呼ばれることもある。

CPMでは、作成したネットワーク図を用いて、各作業の開始と終了の最も早い時期と最も遅い時期を求める。これによって、どの活動がクリティカルパスかを明らかにすることができる。クリティカルパス（経路）とは、必要な時間を積算したときに最長となる一連の活動である。これによって、プロジェクト完了までにかかる最短時間を決定できる。クリティカルパスには余裕時間が全くないため、クリティカルパス上の作業に遅延が発生すると、プロジェクト完了予定日に直接影響する。この方法により、プロジェクトマネジャーは作業の依存関係を考慮しながらスケジュール作成や優先順位付けを行うことができる。スケジュールを短縮させる必要がある場合に、先行作業の完了待たずに次の作業を開始するファストトラッキングや、クリティカルパスにさらにリソースを投入して開発期間を短

縮させるクラッシングの検討を行う場合にも活用できる。

本研究では、変動に対するコントロールを行う際に重要となる工数の再見積り手法において、これらのファストトラッキングやクラッシングを考慮した見積り手法を提案する。

(3) WBS

WBS (Work Breakdown Structure) とは、PMBOKによると、「プロジェクトの成果物や作業をより細かくわかりやすい構成要素に分解すること」である。成果物や作業を「管理できるレベルまで分解していく」ことが、WBS作成の目的である。管理できる最小の単位を「ワークパッケージ」と呼ぶ

本研究では、一つのWBSの所用時間が数時間や一日以内の詳細な粒度のWBSを基準とした、工数計画や進捗、コスト（工数）の変動可視化のための手法を提案する。

3.5. 進捗管理手法

3.5.1. EVM

EVM (Earned Value Management) 手法は、プロジェクトマネジメントにおいて進捗状況の把握、管理を行う手法の一つである。作業の状況を金額などの価値に換算したEV（Earned Value：出来高）という概念で把握する。進捗状況（出来高、スケジュール）とコストを一体として統一的な尺度で把握することができ、計画と実績の差異や変動状況、傾向の分析、コストの予測等に利用することができる。以下に詳細を示す。

3.5.2　EVM による進捗やコストの把握

プロジェクトを計画通りに推進するためには、モニタリングとコントロールが重要である。EVMは、プロジェクトを出来高（EV: Earned Value）とコストの両面で統一的に管理することにより、プロジェクトの進捗やコストの状況を把握する手法である。

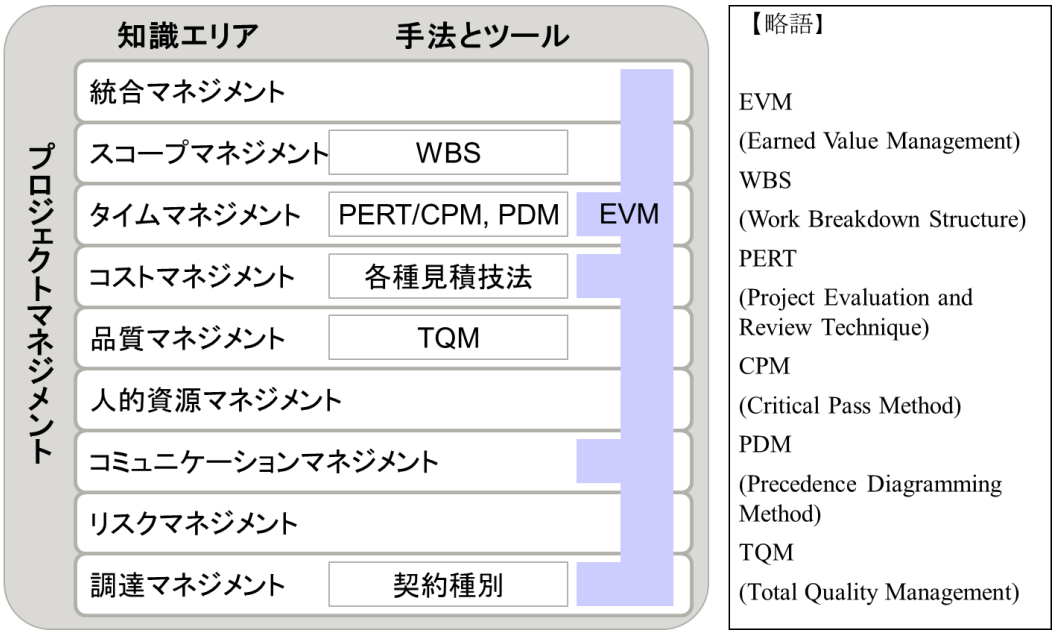


図 3-7　プロジェクトマネジメントにおける EVM の位置付け

図 3-7に、PMBOKの知識エリアや手法との関係を示す。図のように、コミュニケーションマネジメントの知識領域のほか、タイムマネジメント、コストマネジメント、調達マネジメントとの関係がある。また、IPA（2004）によると、EVMは、スコープ（範囲）、スケジュール、コストなどプロジェクトの計画をベースラインとして設定し、そのベースラインを基準として状況をモニタリング及び分析するという視点から、プロジェクトマネジメントの基本原則を取り扱う統合的な性格を持ち合わせている。

EVMは、1967年に1967年に米国防総省の調達規則の一部として制定されたC/SCSC (Cost/Schedule Control System Criteria) が元となっている。その後、1993年に、「クリントン・アドミニストレーション」として国家赤字の見直しが行われ、その際に国防省においてもプロジェクトのパフォーマンス測定や報告手法が見直され、調達規格が「国防省調達規則5000.2-R」として改訂された。これがEVMの始まりである。1998年6月には米国ANSI/EIA #748-98「アーンドバリュー・マネジメントシステムに対する産業界のガイドライン」が公開され、政府調達のプロジェクトの前提条件となっている。英国ではAPMガイドライン、オーストラリアでは、オーストラリア規格AS4817－2003がEVMの規格として相次いで発行されている。日本においても、2003年に経済産業省情報処理振興事業協会により「EVM活用型プロジェクトマネジメント導入ガイドライン」が公開されるなど、EVMの導入が推奨された。これを受け、EVMを導入している企業が増加した。図 3-8に、EVMの概念図を示す。

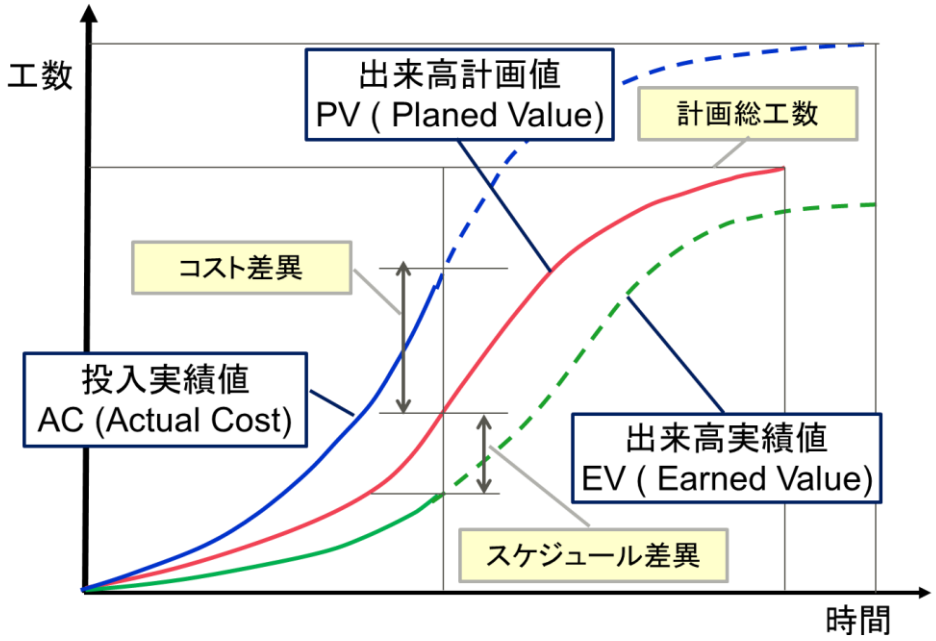


図 3-8　EVM (Earned Value Management) 手法の概念図

EVM では、図 3-8 に示す PV（Planded Value : 計画値）、AC（Actual Cost : 実績コスト）、EV（Earned Value : 出来高）の値を基礎として、現在のプロジェクトの状況分析と将来の予測を行う。各作業に対応した価値を金額等で表したものである。EV は、その作業を実施したことによって得られた価値であり、作業完了時には設定されていた PV 値と同じになる。例えば、PV が 10 の作業を 20 のコストをかけて完了しても、EV は 10 となる。AC とは、その作業を行うにあたり、実際に必要となったコストの実績値である。

EVM を適用するためには、まず、計画段階においてプロジェクトの作業を管理可能な単位に細分化するめ、通常、WBS (Work Breakdown Structure) を作成する。次に、各作業に対応した価値としてのPV、作業別のスケジュール及びコスト計画を決定する。計画時のPVのスケジュールとコスト計画は同一になる。コスト計画を作成するためにはプロジェクトメンバーを確定する必要があるため、プロジェクトの実施体制を計画する。プロジェクトのモニタリング時は、作業の状態に対応したEVとACを計上する。

表 3-7に、これらのEVMで使われる主要な値や指標の一覧を示す。EVMでは、このようにPV、EV、ACの三つの基本尺度を用いて、スケジュールとコストの進捗を計画、把握する。

現時点の状況を分析する指標として、SV（Schedele Value : スケジュール差異)、CV（Cost Variance : コスト差異）、SPI（スケジュール効率指数）、CPI（コスト効率指数）がある。将来の予測に関する値として、EAC（完了時コスト予測）、ETC（残作業コスト予測）、VAC（完了時コスト差異）等がある。

表 3-7　EVM の主要な用語

	略語	正式名称（日本語）	意味
1	WBS	Work Breakdown Structure	プロジェクトの作業を、スケジュールと進捗が把握可能な単位まで詳細化し、階層構造で表したもの。
2	EVM	Earned Value Management	プロジェクトの進捗やパフォーマンス、予測などを、出来高の価値（通常は金額換算）によって把握・管理する方法。
3	BAC	Budget At Completion （完了までの予算）	完了までの予算もしくは予定コスト。
4	PV	Planned Value （出来高計画値）	現時点までに完了した作業に対して、割り当てられていた出来高（価値）のこと。
5	EV	Earned Value （出来高実績値）	現時点までに完了した作業に対して、元々割り当てられていた出来高（コスト）のこと。
6	AC	Actual Cost （コスト実績値）	作業を行うために実際に要したコスト。
7	SV	Schedule Variance （スケジュール差異）	SV ＝ EV － PV ある時点における計画値と出来高の差異であり作業をスケジュール面から見た差異を示す。
8	CV	Cost Variance （コスト差異）	CV ＝ EV － AC ある時点における出来高と実績コストの差異を表す。
9	SPI	Schedule Performance Index （スケジュール効率指数）	SPI ＝ EV ／ PV 各作業のスケジュール面から見た効率を示す。
10	CPI	Cost Performance Index （コスト効率指数）	CPI ＝ EV ／ AC 各作業のコスト面から見た効率を示す。
11	EAC	Estimate At Completion （完成時総コスト）	現時点で見積った完成までの総コスト。 計算方法には、下記の複数の方法がある。 EAC ＝ AC ＋ (BAC－AC) ／ CPI EAC ＝ AC ＋ (BAC－AC) ／ CPI × SPI

（出典：情報処理振興事業協会「EVM活用型プロジェクト・マネジメント導入ガイドライン」を元に加筆、修正）

主な指標について、以下に説明する。

(1) SV (Schedule Variance)

SVは、ある時点における計画値と出来高の差異であり、作業をスケジュール面から見た差異を示す。値がマイナスの時は、計画よりもスケジュールが遅れていることを意味する。

(2) CV (Cost Variance)

CV は、ある時点における出来高と実績コストの差異を表す。この値がマイナスの時は、予定よりもコストが増加していることを意味する。

(3) SPI (Schedule Performance Index)

SPI < 1の場合は、進捗の効率が予定よりも低いことを意味する。

(4) CPI (Cost Performance Index)

CPI > 1の時は、生産性の効率が予定よりも低いことを意味する。

(5) EAC (Estimate At Completion)

完成時の総コストの予測値である。

これらの指標を活用することにより、計画と実績の差異の把握や予測を定量的に行うことが可能となる。ただし、木野（2011）が指摘しているように、詳細な分析を行ったとしても、計画以上の精度で結果を得ることはできない。したがって、計画の質を高めることが必須であるとされている。

また、プロジェクトは、「個別性」の特性により計画段階においては不明確な点が多く、「不確実性」の特性により実施段階において予定外の出来事や変更が発生することが多い。このため、プロジェクトの実施段階においては、スケジュール変更を余儀なくされることが頻繁に発生する。スコープやスケジュールが変更となれば、それらに伴い体制の変更が必要となることもあり、必要に応じて計画の見直しを行うことが必要となる。

なお、木野（2011）は、「EVMの各種の値は、差異や傾向を表す指標であり、傾向が悪いことがわかっても、何が問題であるかまでは示されない。」と述べている。

本研究では、一つのWBSの粒度を数時間程度の詳細なものとすることで、EVMの適用方法を拡張し、日々の進捗や工数の変動の可視化と、スケジュール遅延等の原因分析への活用を可能とするための方法について提案する。この際、工数計画の作成を自動化する仕組みを取り入れることによって、スケジュールやスコープの変動が発生した際の計画の見直しを容易に行えるようにする。

3.6. プロジェクトの変動マネジメントに関する先行研究と課題

本節では、プロジェクトの変動マネジメントに必要なプロジェクトの状態の計測や監視（モニタリング）、予測、コントロール及び再見積りの観点での先行研究と課題について述べる。

プロジェクトの計測及び監視の方法については、スコープ（進捗）やコスト及びスケジュールを一体として定量的に把握する方法として、前述のEVMが普及している。Fleming & Koppelman (2003) は、EVMを「プロジェクトのコストマネジメントに必要不可欠な技法」としている。PMBOK (PMI 2017) やKerzner (2009) は、EVMをプロジェクトのスケジュールとコストのマネジメント技法として紹介している。EVMをスケジュールマネジメントに用いた事例が、Vandevoorde and Vanhoucke (2006)、箱嶋（2007）、星ほか（2007）など、数多く報告されている。他方、コストマネジメントについては、プラント建設の分野において、齊藤・鈴木（2013）が単価と数量の概念をEVMに追加することによるコストマネジメントの方法を提案している。

EVMのソウトウェア開発への適用方法としては、EVMを活用したスコープ（進捗）やコスト及びスケジュールの定量的把握方法について、Anbari F.(2003)、Fleming and Joel (2000)、Lipke et al (2009)、Cooper et al. (1993) 等、多数の研究がある。Vandevoorde and Vanhoucke (2006) は、EVMを期間の変動の定量化に活用する方法を提案している。PMBOK第6版では、スケジュールベースのEVM手法が紹介されている。Demirors (2013) は、ソフトウェア開発へのEVMの適用方法と課題について報告している。しかし、前章で述べたように、ソフトウェア開発においては、目に見えにくい作業の計画値や出来高を定量化すること自体が難しいため、計画値や出来高を定量化するための工夫やデータ収集が必要となる。

進捗率やコストを把握するための方法として、前田（2004）や米花ほか（2003）によって、一律の固定比率を用いた簡易的な進捗率の計上方法や、勤務時間の合計値を用いた工数把握の方法が提案されている。ただし、これらの方法は、プロジェクト全体の進捗率やコストの傾向を把握する目的には適しているが、機能別や担当者別の進捗や工数の日々の変動を把握する上では、把握のサイクルや粒度が不十分である。このように、EVMの適用方法としては、プロジェクト全体の傾向を俯瞰的に把握するための利用方法が一般的であり、機能や人別の詳細WBSの粒度で、日次で変動を把握することには適用されていない。

EVMの適用方法とは別に、日々の変動把握については、渡辺・丸山（2009）が、一つのWBSが数時間単位の詳細なWBS単位で工数の計画値と実績値の差異を把握する方法を提案している。Jones (2008) が指摘している、時間単位での精度の高い定量的マネジメントはソフトウェア開発の中で進んでいない領域であるが、渡辺・丸山（2009）の方法を取り入れた上でEVMを適用することができれば、精度の向上を図ることができる可能性がある。

変動の把握を詳細な粒度で精度良く定量的に把握することができれば、スケジュールやコストの悪化などの問題発生を早期に検知、予測し、実績データを活用することによってプロジェクトを計画通りに進めるための対策を早めに実施することが期待される。ただし、問題発生の原因分析や対策、要員追加のための再見積り等のコントロールの方法は、プロジェクトマネジャーの経験に依存していることが多い。このような、スケジュールの遅延等に対するプロジェクトのコントロールのための手法や、要員追加のための再見積り手法の研究は先行研究では行われていない。

3.3.2節で示した再見積りに関する先行研究は、実績データを用いて見積りの精度を高める方法や、実績データを利用した後工程の工数見積り手法などである。しかし、それらの先行研究では、規模の増加や期間の短縮等による反復回数の減少などの、変動に伴って発生する各種の工数増加要因への影響は考慮されていない。このような、当初の計画を変更する場合のコントロール方法は、経験を積んだプロジェクトマネジャーの実践知として、暗黙知となっていることが多い知識領域であるため、先行研究では扱われていなかったと考えられる。本研究では、プロジェクトマネジャーの暗黙知となっている割合が多いコントロールに関する過去のプロジェクトで獲得された知識を形式知化する手法を提案する。

3.7. プロジェクトマネジメントの知識継承の課題と研究

3.7.1. プロジェクトマネジメントの知識継承の課題

ITシステムの社会や経済における重要性が増す中で､経験が豊富なプロジェクトマネジャーの知識や過去のプロジェクト経験から得られる知識を、いかに組織内で蓄積し、他のプロジェクトマネジャーに継承するかが重要となっている。

PMI (2015) によるプロジェクトマネジメントの知識継承に関するアンケートによると、最も効果的な知識継承を行っている組織では、74%～75%のプロジェクトがスケジュールと予算を目標通りに達成しているが、効果的な知識継承を行えていない組織では、スケジュールや予算の目標を達成したプロジェクトは、42%～48%スに止まっている。ここで、効果的な知識継承を行っている組織とは、知識の特定、獲得、共有、利用、評価のサイクルを全て効果的に実施している組織であると定義しており、逆に、効果的な知識継承を行っていない組織プロジェクトとは、これらのサイクルを実施していない組織であると定義している。このアンケート結果にも表れているように、プロジェクトの成功率向上のためには、プロジェクトマネジメントの知識継承の取り組みが重要である。しかしながら、各企業や組織内でプロジェクトマネジメントに関するガイドや教訓集の作成、教育などのプロジェクトマネジメントの知識継承の取り組みが行なわれているが、なかなかその効果が現れてこないケースや､各種の施策の効果を明らかにしにくいことがある。この背景には、プロジェクトマネジャーに必要な知識が多岐に渡ることや、知識の多くが個人の暗黙知となっていることなどが背景にあると考えられる。特にソフトウェア開発プロジェクトは、仕様の追加に伴う規模増加や、人に依存する生産性の変動等の様々な要因によって、計画に対して規模や進捗、コスト等の変動が発生する。さらに、ソフトウェア開発の成果物はドキュメントやソースコード等であり、開発作業には成果物には現れない検討作業やレビュー等の様々な作業が含まれるため、規模や進捗、作業の状態等の正確な把握が難しい。このため、ソフトウェア開発の変動に対するコントロールは、プロジェクトマネジャーの経験に基づく暗黙知に依存する部分が大きいと言える。このため、必ずしも継承すべき知

識の範囲等が明確になっていない状況で、その都度、必要と考える知識継承に取り組む必要があり、その効果を明らかにしにくい面があると言える。

以下では、プロジェクトマネジメントの知識の種類や体系について外観し、従来のプロジェクトマネジメント知識の継承に関する先行研究や取り組みについて述べる。

3.7.2. プロジェクトマネジメントの知識体系

プロジェクトマネジメントは、プロジェクトマネジャー個人の力量や経験に依存する部分が多いため、プロジェクトマネジメントに関する知識や技法を整理し体系化することで、プロジェクトマネジメントに必要な知識を他のプロジェクトマネジャーが共有し、プロジェクトマネジメント力の向上を図るための取り組みが進められてきた (Cicmil and Hodgson 2006) 。以下に主なプロジェクトマネジメントの知識体系の概要を説明する。

(1) PRINCE

1989 年に、英国CCTA (the Central Computer and Telecommunications Agency) が開発したPRINCE (Project in Controlled Environment) は、英国の情報システムのプロジェクトマネジメント標準として英国政府機関で利用され、IT分野でも使われるようになった。1996年に、より汎用的な手法として、PRINCE2 (Project in Controlled Environment, 2nd ver.) が出版され、現在PRINCE2 は第5版が発行されている (TSO 2009) 。PRINCE2 は、英国以外にも利用が広がっている。

(2) PMBOK

米国においても、1996 年に設立されたPMI (Project Management Institute) によって、プロジェクトマネジメントの知識体系としてPMBOK (Project Management Body of Knowledge) が出版され、現在は第6版が出版されている (PMI 2017) 。PMBOK では、プロジェクトマネジメントに必要な10 個の知識エリアを規定している。当初のPMBOKは、ウォーターフォール型の開発モデルを基本としていたが、Callegari (2004) やCallegari and Batos (2012)による統一プロセスなどの他の開発モデルとの融合に関する研究結果が報告されており、PMBOK第6版では、アジャイル開発のプロセスについても取り込まれている。

(3) ISO21500

あらゆる製品やサービスについてグローバル化が進んでおり、これに伴ってプロジェクトマネジメント分野での国際標準化も進められている。

ISO21500 (ISO21500:2012) とは、2012年9月1日に発行されたプロジェクトマネジメントに関する国際標準である。標題は「Guidance on project management（標題訳：プロジェクトマネジメントの手引）」となっており、プロジェクトマネジメントに関する包括的なガイド

ラインとして、ガイダンス規格となっている。ISO21500の内容は主に、プロジェクトマネジメントの概念と、プロジェクトマネジメントのプロセスに焦点が当てられたものである。ISO21500の制定にあたっては、PMBOKを開発。運営しているPMIも参加している。

(4) P2M

P2M (Program & Project Management) は、1998年から経済産業省とエンジニアリング振興協会のプロジェクトマネジメント導入開発委員会により開発され、日本型のプロジェクトマネジメント知識体系としてまとめられた (PMAJ 2014) 。

それまでの主要なプロジェクトマネジメントの体系は、建設やITシステム構築等を想定したものであったが、P2Mでは、イノベーション推進と価値創造に対応できるプログラムやプロジェクトマネジメントへの適用を想定し、「外部環境の変化を意識したうえで、複雑な使命に問題解決の道を開き、事業価値を向上する」ことを目的としている。また、P2Mのプログラムとは、「全体使命を実現する複数のプロジェクトが有機的に結合された事業」であり、知識創造動態モデル野中（2010）の考え方が取り込まれている。特に「バリューマネジメント」の「価値の源泉」に関して、プロジェクトで生み出された様々な知識を蓄積し活用する必要性について言及されている。P2Mは、時代と環境の変化に合わせて二度に渡り改訂されており、2015年4月に改定3版が発行された。プログラムマネジメントの充実と、プロジェクトマネジメントのマネジメント要素をISO21500シリーズへの合致等が行われている。

3.8. システム開発プロジェクトの知識継承の研究

プロジェクトマネジメントの知識継承のための研究としては、知識の内面化のための Hawk (1999) などの学習に関する研究や、そのための知識の表出化に関する研究等が行われている。

企業のシステム開発プロジェクトにおける知識継承の実践的な取り組みの研究として、内田ほか（2010、2013）は、過去の事例からの経験知識の表出化とケースメソッドによる知識の内面化の仕組みを提案している。具体的には、ポストプロジェクトレビューにおいて、様々な阻害要因を排除し、本質的な知識を抽出するための原因分析手法を提案し、活用できる形式として知識化様式を定義している。また、事例から抽出された知識に基づいてケースメソッドを設計し知識の受け手が実践で使える知恵とするための、内面化を支援する方法を提案している。内田ほかの手法は、企業内の情報システムの開発プロジェクトでの適用実績だけでなく、プラントなどのシステム開発においても適用実績がある。

3.9. 研究開発プロジェクトにおける知識継承の研究

研究開発プロジェクトは、情報システムのプロジェクトと異なり、期間が長い点やマネジャー個人が経験できるプロジェクトも限られる点が特徴である。内平（2010）は、研究開発プロジェクトのマネジメント知識を組織内や世代間で継承するための知識継承フレームワークを示している（Uchihira 2005, 内平 2010, Uchihihira 2012）。提案されている知識継承フレームワークは、終了したプロジェクトのマネジャーによる知識の送り手としての知識の表出化と、現在進行中のプロジェクトのマネジャーによる知識の受け手としての知識の内面化から構成される。具体的な手法として、ポストプロジェクトレビューにおいて、研究開発マネジメントに固有の知識構造であるフェーズレビューのチェックポイントを用いて終了プロジェクトを分析し、ケースとして表出化する「構造化プロジェクト分析手法」を提案している。進行中のプロジェクトのフェーズレビューにおいては、ケースを活用して将来のチャンスとリスクを創出する「内面化ワークショップ」を提案している。

3.10.先行研究のまとめと本研究のアプローチ

3.10.1. プロジェクトの変動マネジメント手法の必要性について

従来の定量的マネジメントに関する研究は、メトリクス自体の定義や精度の向上に関する研究が中心であった。3.2.2節の再見積りに関する先行研究も、実績データを利用して見積りの精度を向上させるものであった。しかし、プロジェクトマネジメントにおいて課題の多いプロジェクトの変動の把握やコントロールの方法や、計画変更や要員追加にともなって発生する工数増加を考慮した再見積りの方法等に関する知識をいかに継承するかについては研究が行われていない。そこで、本研究では、プロジェクトを計画通りにマネジメントする上で重要な変動の監視と制御に焦点を当て、早期の原因分析やコントロールを可能とするため、詳細な粒度で日々の変動を可視化する手法と、知識の活用や継承を促進するために知識を形式知化する方法を提案する。

工数見積りに関する従来の研究の多くは、計画立案のための工数見積りの精度向上のための研究であった。再見積りについての研究も多数あるが、工程の情報から次工程以降の工程別の工数を見積もることによる見積り精度向上を目的としたものである。プロジェクトの変動発生に対して要員数やスケジュール等を変更する際に必要となる工数の再見積りには、規模の変動や計画の変更自体が期間短縮や要員追加、作業分割などの工数増加に及ぼす影響などはプロジェクトマネジャーの経験に基づいた暗黙知となっており、こうした開発期間や作業分割等を考慮した再見積りの研究は行われていない。

本研究では、従来はプロジェクトマネジャーの経験や暗黙知に依存していた、規模や期間、作業の反復回数の変更があった場合の工数や要員数の再見積りの方法を含む、コント

ロールの知識を形式知化する方法を提案する。

本研究で提案するプロジェクトの変動に関するマネジメント手法は、規模や進捗、コスト等の変動を可視化する方法のシステム化と、変動に対するコントロール方法に関する知識のパターン化や形式知化の両面によって知識の継承を促進するためのものである。すなわち、提案する手法を知識継承の観点で捉えると、プロジェクトマネジメントの知識を、システムやパターンとして形式知化する手法として位置付けられる。

3.10.2. プロジェクトマネジメントの知識のモデル化の意義について

従来の知識継承に関する研究は、知識の表出化や継承のためのプロセスの解明や、知識の共有や継承の方法に関するモデル化等が中心であった。

他方、プロジェクトマネジメントの知識の蓄積や活用を目的とした具体的な知識の体系化については、PMBOK や PRINCE などの知識体系として整理されている。しかし、Horner and Yong (2006) は、「PMBOK ガイドは、形式知や宣言的な知識に偏っており、暗黙知や理由（Why）に関する知識にはあまり注意が払われていない」と分析している。例えば、プロジェクトの変動に対する原因究明や対処方法については、プロジェクトマネジャーの暗黙知の領域と見なされていた部分が多く、従来の知識体系化にはほとんど含まれていない。また、プロジェクトマネジメントの知識を継承する手法としての形式知化やシステム化などの観点での知識の分類や整理についての研究は行われていない。このように、従来の PMBOK や ISO21500 などの知識体系が、実施すべきこととしての What の種類を分類するに止まり、どのようにプロジェクトの状況を監視、コントロールするのかという、How や Why に関する知識を蓄積及び活用する上では不足している。

本研究では、開発プロジェクトの変動の監視と制御に関する知識の具体的な内容に焦点を当て、制御対象としてのプロジェクトに対する監視（モニタリング）と制御（コントロール）及び、プロジェクトの状態を表す管理要素とそれを介した変動管理、さらにそれぞれの背景知識となる実践知という形の知識レイヤーに分けて分類し、各知識レイヤーや具体的な知識要素間の動的な関係を示す。これにより、プロジェクトマネジメントの知識を、どの場面で、どの知識と組み合わせて活用するのかなどが明確になるため、知識の継承や、計画的な知識の蓄積、システム化可能な領域の見極めなどへの活用が進めやすくなり、プロジェクトマネジメントにおける知識の継承の実用的な枠組みとなることが期待される。

このモデルの活用方法や効果を、プロジェクトの変動を可視化する手法を適用した事例によって検証し、プロジェクトマネジメントの知識の蓄積や継承および活用のための研究の発展に貢献する。

知識構造モデルの体系化に当たっては、内平（2010）による研究開発のプロジェクトマネジメントの知識継承に関する知識の分類や構造化の方法を参考に、ソフトウェア開発のプロジェクトを制御対象として捉え、監視と制御に関する具体的な知識を構造化し、知識

階層間の動的な関係等を考慮したモデルとして提案する。

また、提案する知識構造モデルの発展的な活用方法として、プロジェクトマネジメントの知識を補完するためのシステムの構築や、AI 技術のプロジェクトマネジメントへの活用に関する検討への活用方法について考察し、知識構造モデルの有効性を示す。

4. ソフトウェア開発プロジェクトの変動マネジメント手法

本章では、過去のプロジェクトマネジメントの知識の活用や継承を促進するための方法の一つとして、ソフトウェア開発の変動マネジメント手法について提案する。最初に、ソフトウェア開発のプロジェクトマネジメントの知識の種類と知識の間の関連性を整理した上で、提案する変動マネジメント手法がどのような知識領域を対象としているかを示す。変動マネジメント手法の有効性については、次章で、プロジェクトへの適用事例によって検証する。その後、プロジェクトの変動マネジメントに関する知識をさらに一般化した知識構造モデルを提案し、知識の形式知化や活用方法の研究の枠組みとする。

なお、本研究で対象とするプロジェクトは、企業内や企業間における業務システムの開発を対象としている。開発プロセスとしては、ウォーターフォール型の開発プロセスモデルのように、特定の目的を達成するためのシステムに対する要求を元にして、要件定義から、設計、構築、テスト、運用開始までの全体のスケジュールやコストの計画を策定し、各工程を計画通りに推進することを基本として開発を推進する開発手法を前提とする。

4.1. プロジェクトマネジメントの知識の活用、継承について

社会や企業における IT システムの重要性が増す中で、システム開発のプロジェクトマネジメントの知識をいかに継承するかが各組織における重要な課題となっている。ソフトウェア開発プロジェクトの場合は、第 2 章で述べたように、規模の増加や、人に依存する生産性の変動等の様々な要因によって、計画に対して規模や進捗、コスト等の様々な変動が発生するが、それらの状態の正確な把握が難しい。変動の把握が不正確な場合や管理の粒度が粗い場合は、プロジェクトマネジャーが進捗遅延などの問題に気付くのが遅れ、コントロールに支障をきたすことがある。このため、ソフトウェア開発においては、変動をいかに早期に正確に把握し、原因分析に基づいて適切なコントロールを行うかが重要である。経験の豊富なプロジェクトマネジャーは、変動を的確に把握し、早期に問題点とその原因を見出し、問題が拡大する前に対策を構ずるが、そのために必要な知識は多岐に渡り、個々のプロジェクトマネジャーの暗黙知となっていることが多い。したがって、過去のプロジェクトマネジャーの経験に基づく知識を、他のプロジェクトマネジャーが活用し、組織内で効果的な知識継承を行うためには、プロジェクトマネジメントの知識が、具体的にどのような要素から成っているかを整理する必要がある。そこで、プロジェクトマネジャーが果たす役割を表したプロジェクトマネジメントの概念図を図 4-1 に示す。

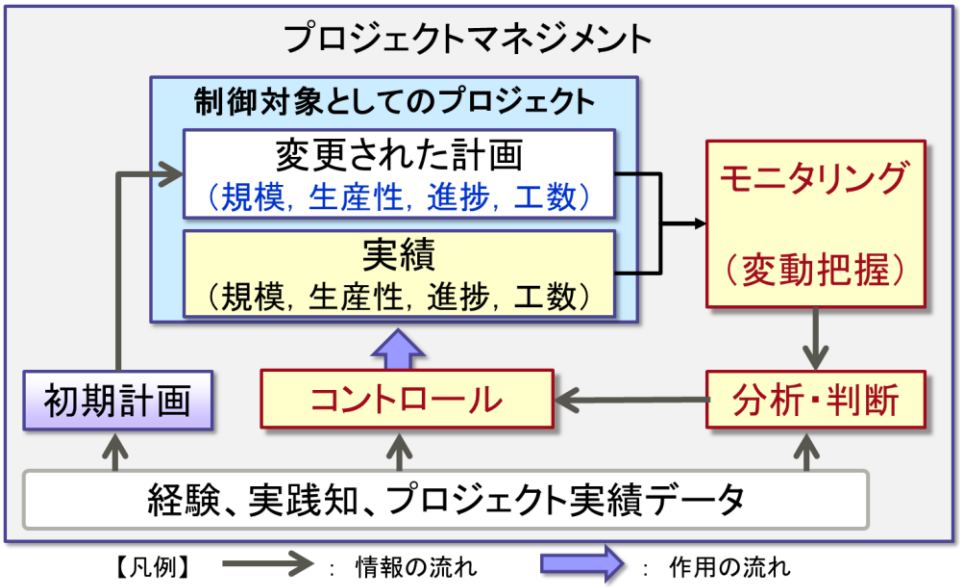


図 4-1　プロジェクトマネジメントの概念図

プロジェクトマネジメントは、計画と実績の差異を監視し、その差が少なくなるようにプロジェクトを制御（コントロール）する活動と捉えることができる。図 4-1 における「制御対象としてのプロジェクト」は、ソフトウェアなどの成果物や要員、コスト、期間等の制御の対象としての各種リソースの集合を意味している。図では、計画段階で設定した見積り又は目標としての規模、生産性、進捗（スケジュール）、工数等を基準として、計画と実績の差異（変動）をモニタリングし、差異が少なくなるようにコントロールするという一連のプロジェクトマネジメントのプロセスを示している。プロジェクトマネジャーは、計画作成、モニタリング、分析・判断及びコントロールのそれぞれのプロセスにおいて、過去の経験や既存の知識を活用しながらプロジェクトマネジメントを実行する。本研究では、上記のように、プロジェクトを制御対象と見なし、プロジェクトの変動を把握して計画と実績の差を少なくなるようにコントロールする活動を、プロジェクトマネジメントと捉え、そのためのプロジェクトの変動に対する監視と制御の手法を提案する。

この際に活用する知識がいかなる要素から構成されているかを明らかにし、暗黙知となっている知識を形式知化やシステム化することによって、プロジェクトの変動に対するマネジメントの知識を継承する方法を提案する。

以下では、変動に対するプロジェクトマネジャーの知識の要素や構成を整理した後に、形式知化やシステム化によって、知識継承を促進する方法を提案する。

4.2. プロジェクトの変動マネジメントに関する知識関連図

図 4-1 に示すプロジェクトマネジメントのプロセスにおいて、プロジェクトマネジャーが、どのような情報を元にプロジェクトの変動を把握（モニタリング）し、いかなる知識を活用して問題の原因分析やコントロールを行うのかを図 4-2 に示す。

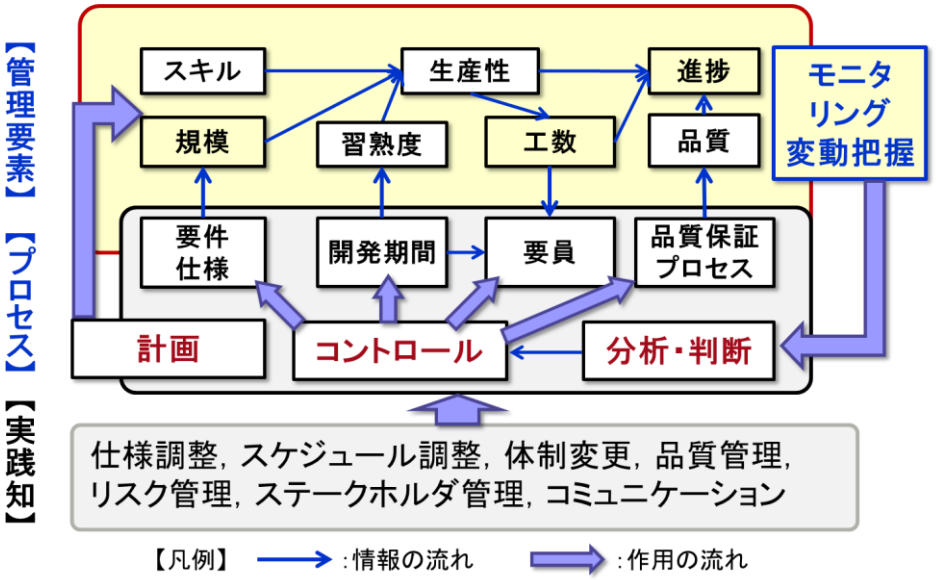


図 4-2　変動マネジメントの知識関連図

図 4-1 では、プロジェクトの変動に対するマネジメントの知識を「管理要素」、「プロセス」、「実践知」の 3 つの階層に分けている。ここで、「管理要素」という知識の階層を定義した理由は、プロジェクトをコントロールする上で、計画に対する実績の差異の把握（モニタリング）に基づく原因分析や対処方法の判断において、規模、生産性、進捗、コストなどの主要なメトリクスや、進捗に影響を及ぼす生産性、要員スキル、習熟度等の各種の要因に関する知識を駆使することが必要となるためである。次に、計画策定、変動把握のためのモニタリング、分析及び判断、コントロールの一連のマネジメントプロセスについて、各知識要素間の関係や知識の階層関係を示している。「管理要素」に含まれるソフトウェア規模、工数、進捗、品質等の要素については、各管理要素間の関係性を示している。これは、各管理要素間の関係性に関する知識が、原因分析やコントロールを行う上で重要となるためである。変動が発生した時に、プロジェクトマネジャーが適切なコントロールを行うためには、変動が発生している様々の管理要素の間の関係性を考慮しながら、直接的及び間接的な原因を把握し、要件や仕様、作業手順、人などのうち何が原因かによって適切な対策を実施することが必要であり、このためには、各管理要素間の関係に関する知識が重要である。なお、「プロセス」に関する知識には、モニタリング結果に対する分析や判断と対処方法を含めている。最後に、コントロールや分析・判断を行う際に活用される、

過去のプロジェクトの経験等に基づいた暗黙知を含む知識を、「実践知」として分類した。

プロジェクトマネジャーは、計画策定時には、ソフトウェアの規模を見積もり、過去の類似プロジェクトの実績データを参考にして工数を見積もり、計画を策定する。実行段階では、進捗や工数をモニタリングし、計画と実績の差異の原因を分析する。この分析結果に基づいて、対処方法を判断し、仕様調整やスケジュール変更、要員追加等のアクションによって、要求や仕様、開発期間、要員、品質保証プロセス等の修正や改善を行うことでプロジェクトをコントロールする。この際、経験の豊富なプロジェクトマネジャーは、定量的なデータに加えて過去の経験に基づいた判断を基にプロジェクトをコントロールする。また、スケジュール変更や要員追加を行う際は、生産性に及ぼす要員のスキルや習熟度、開発期間等の影響度合いを考慮し、過去の経験を元に工数の再見積りを行う。このような一連のプロジェクトマネジメントの知識は、プロジェクトマネジャーに依存した暗黙知となっていることが多く、他のプロジェクトマネジャーへの知識継承は難しかった。これに対して、図 4-2 のように知識の種類別に階層化し、知識要素の間の関係性を明らかにすることによって、計画策定やモニタリング、コントロールの各場面において、どのような情報を元に、いかなる知識を組み合わせ、どう対処すべきかがわかりやすくなり、蓄積すべき知識も明確になる。

本章では、プロジェクトマネジメントの知識の活用や継承を目的とし、知識の形式知化やシステム化を組み合わせた変動マネジメント手法について提案する。この手法は、下記の 3 種類の手法から構成される。

(1) 進捗・コストの変動可視化手法（デイリーEVM 手法）

(2) ソフトウェア規模の変動可視化手法

(3) 変動に対するコントロール手法

ここで、図 4-2 の知識関連図を用いて上記の各手法の位置付けを示す。(1)の進捗及びコストの変動可視化と(2)のソフトウェア規模の変動可視化の手法は、知識関連図における、計画に対する進捗、工数や規模の変動を把握するモニタリングプロセスに対応する。(3)の変動に対するコントロール手法は、知識関連図において、モニタリングの結果を分析し、原因に応じて要件・仕様、開発期間、要員、品質保証プロセス等に対するコントロールを行うプロセスに対応する。

4.3. 進捗・コストの変動可視化手法（デイリーEVM 手法）

初めに、進捗及びコストの変動可視化のための手法を提案する。規模の変動可視化の手法については、把握のサイクルやプロセスが進捗やコストの可視化とは異なるため、次節で述べ、その後、変動に対するコントロール手法について提案する。

プロジェクトの変動を把握するための方法としては、主に、進捗とコストの定量的な把握によって行われることが多い。この際のデータ収集や分析には運用負荷がかかるため、実効性を考慮した現実的な方法として、週や月の管理サイクルで、サブシステムやチームを単位とした粒度で進捗やコストを把握する方法が一般的である。このような俯瞰的な監視方法のみでは、第 2 章と第 3 章で述べたように、機能別や人別の作業遅延等を把握することは難しく、問題の発生に気付くのが遅れ、原因究明や対策が遅れることによって問題が拡大してしまうリスクがある。経験が豊富なプロジェクトマネジャーは、メンバーの作業状況やコミュニケーションなどの情報から問題発生の予兆を捉えたり、経験上、問題が発生する可能性の高い作業や担当者の状況を重点的に確認したりすることによってコントロールを行っている。しかし、経験の浅いプロジェクトマネジャーは、問題が発生しやすい作業やその徴候などの見極めができず、問題の予兆に気付かずに対処が漏れてしまう可能性がある。経験の浅いプロジェクトマネジャーでも変動を的確に把握できるようにするためには、収集するデータの粒度をなるべく詳細な粒度とするのが望ましい。理想的には、機能や人別の詳細な WBS の単位で、日次の変動を把握できれば、早期の異常発見と、詳細 WBS 別の実績データに基づいた的確な原因分析が可能となると期待される。これによって、経験の豊富なプロジェクマネジャーも含めて問題の察知や原因分析を効率よく行え、効果的なマネジメントが可能となる。さらに、変動の可視化を行う方法やプロセスを標準化しておくことにより、原因分析や対処方法などのノウハウを形式知として蓄積し、知識継承に活用することが可能となる。

本研究では、機能別や担当者別の詳細な WBS 単位での進捗やコストの変動を、日次で可視化し、得られた変動情報と過去の経験等の知識を活用して円滑にプロジェクトをマネジメントするための手法を提案する。この際、詳細な粒度で変動を把握する上で阻害要因となる運用負荷を増やすことなく適用可能とするため、各種パラメータを元に工数計画を一括作成する方法や、実績工数を元に進捗率や出来高を自動計上する仕組みを検討する。さらに、過去の人のノウハウをパターン化することで形式知化し、形式知化された知識を活用することで、変動の原因に応じた適切なコントロールを可能とする。これにより、精度向上と運用負荷低減の両立を図り、過去の知識や実績データを活用可能な変動マネジメント手法の実現を目指す。

4.3.1. ソフトウェア開発の進捗・コスト管理の課題

プロジェクトで発生するスケジュール遅延等の各種問題に対して、作業時間や要員の調

整等のコントロールを適切に行うためには、成果物の量（出来高）とコストを一体とした把握が重要である。しかし、第 3 章 3.6 節で述べたように、既存の俯瞰的な EVM の適用方法等は、プロジェクト全体の傾向を予測して対処方針を検討することには役立つが、日々の問題の察知や原因究明等に活用することは難しい。

プロジェクトの円滑なマネジメントのためには、スケジュールの遅延やコスト増加を早期に察知し、原因を分析して問題を拡大させないための対策を講じることが重要である。このためには、進捗や工数の変動を短いサイクルで把握し、詳細な管理粒度で変動を定量的に可視化することが求められる。しかし、詳細な粒度のWBSを元にした計画策定や、データ収集、集計のため運用負荷が高くなると実行することが困難である。

EVM を適用する際の出来高については、計画出来高（PV）としての計画工数に進捗率を乗じて算出ずる方法が一般的な方法として提案されている。このためには進捗率の把握精度をいかに高めるかが課題となる。進捗率の把握方法には、担当者から進捗率を報告させる方法、成果物の量を元にした方法及び前述の固定比率計上法等があるが、いずれも主観性や曖昧性が内在し、精度の観点で課題がある。また進捗や工数の定量的な把握のためには情報収集に負荷が掛かるため、ウォーターフォール型開発モデルのプロジェクトでは、EVM の適用においても、週や月の管理サイクルで、サブシステム単位やチーム単位の進捗やコストのデータを集計し把握するのが一般的である。このような管理サイクルや粒度では、プロジェクト全体の傾向を俯瞰的に把握するための活用に止まり、収集したデータをタイムリーな問題把握や原因究明などに活用することはできていないことが多い。

さらに、変動を正確に把握するためには、ベースラインとなる工数計画立案や実績データの収集方法も重要となる。しかし、実際には各プロジェクトマネジャーの経験に基づいた方法で WBS の作成や実績データ収集を行っていることが多い。このような属人的な方法では、ベースラインとなる計画工数や実績データの粒度がプロジェクトによって異なり、見積りや計画立案に過去のプロジェクトの統計データ等を活用することも難しくなる。

前述のように、工数計画や実績収集の単位である WBS の粒度が粗い場合には、計画と実績の差異が生じた原因がプロジェクトの実行における問題に起因しているのか、見積りの誤差によるものなのかもわからなくなる。これに対して、詳細な粒度で日々の変動状況を定量化できれば、早期に異常を把握し、発生した問題の原因究明等に定量的な変動状況のデータを活用することが可能となる。また、WBS や実績データ収集の粒度を標準化することにより、過去の統計データや知識の活用が可能となると考えられる。このための参考となる方法の一つとして、東ほか（2003）によってワークパッケージレベルの WBS を単位とした進捗及び工数の把握によって EVM を適用する方法が提案されている。ただし、その粒度は「外部設計書作成」、「コーディング」、「単体テスト」、「結合テスト」や「レビュー」といった工程の区切りとしての単位であり、一つのワークパッケージの作業工数が数日や数週間となる場合がある。このため、日々の変動把握や遅れの原因究明等に活用する上では粒度が粗い。また、進捗率の把握方法は設計書の枚数や指摘件数を元に計算する方法で

あるため、進捗率の精度にもばらつきが生ずる可能性がある。データの収集については報告書を元にした方法であるため、実績収集の負荷が高い。

また、工数計画作の際には、WBS の洗い出し、工数計画の作成に加え、機能別や担当者別の WBS ごとのドキュメント枚数や工数、開始日、終了日及び作業時間等を個別に見積もる必要があり、計画作成の負荷も大きい。このような計画作成は、プロジェクトマネジャーの経験に基づいて行われることが多いが、そのような属人的な方法では計画工数や収集するデータの精度にばらつきが生じ、計画と実績の差異の原因の分析や対策の検討に支障をきたすことがある。

以上のように、詳細な WBS の粒度での日々の変動を、精度と負荷低減を両立させながら把握するための方法は確立されていない。

4.3.2. 進捗・コストの日次での変動可視化手法の概要

前述の課題を解決するための方法として、機能別や担当者別の詳細な WBS の粒度で、日次で変動を可視化する手法を提案する。

工数の把握方法としては、渡辺・丸山（2009）が提案している、作業工数を時間単位（0.1 時間単位）で集計する方法を参考とする。この方法は、WBS 別に「基準時間」を設定し、計画と実績との差異を把握する方法である。ただし、この方法は詳細 WBS 別の時間単位での実績工数の把握方法を中心に提案しているものであり、進捗率と出来高の計上方法や、可視化された変動情報の活用方法、インプット情報の一つである規模変動の可視化及び変動に対するコントロールの手法は含まれていない。

進捗計上の方法は、東ほか（2003）のワークパッケージ単位の進捗率を積み上げて集計する方法を採用する。この際に、一つの WBS を数時間（1 日以内）程度の粒度で設定し、進捗率計上のルールを設定しておくことで、日々の進捗率と出来高を数時間のぶれ幅で把握できると期待される。このような詳細な粒度での進捗や工数の把握を行う上では、工数計画作成やデータ集計等の運用負荷の増加が問題となる。そこで、これらの工数計画の一括作成や進捗率の計上の自動化等の、可視化のための作業負荷を軽減するための仕組みを提案する。工数計画作成の一括作成は、実績データを活用した見直しによる計画の精度向上や、変動に対するコントロールの際の要員追加、スケジュール変更などの計画修正を容易に実行可能とするために重要となる。以下に具体的な実現方法と特徴を示す。

4.3.3. 変動可視化手法の全体像

図 4-3 に変動可視化手法の全体像を示す。

以下では、本研究において提案する変動可視化の手法は、EVM の考え方を日々の詳細 WBS 別の変動可視化に応用するものであるため、「デイリーEVM 手法」と称する。

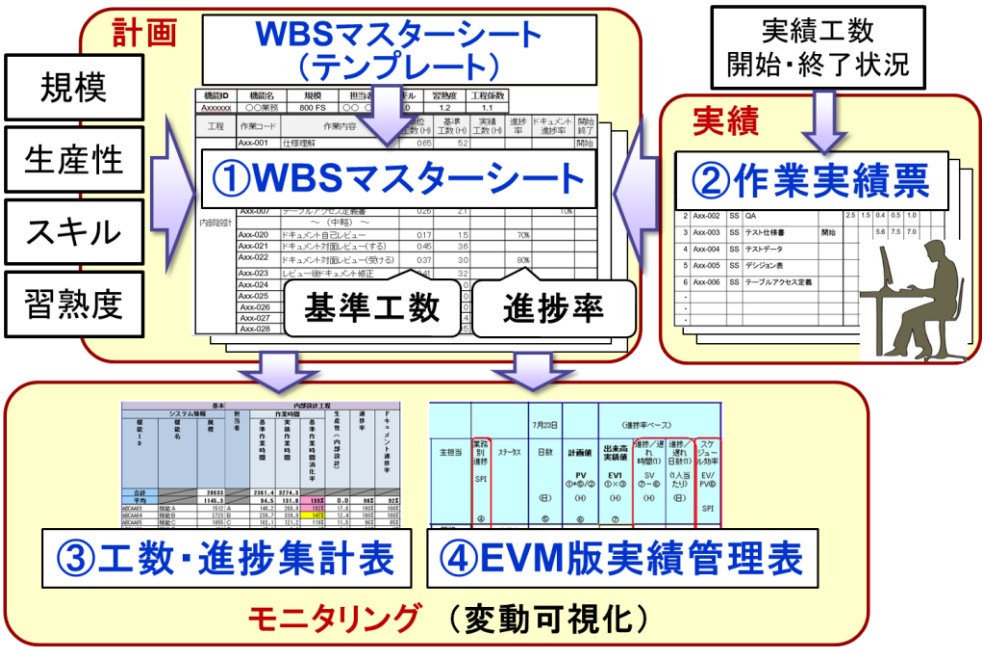


図 4-3　変動可視化手法「デイリーEVM 手法」の全体像

図 4-3 における①から④までのシート類の概要を示す。

計画作成プロセスでは、①「WBS マスターシート」を用いてベースラインとなる WBS 別の「基準工数」の設定を行い、実績収集のプロセスにおいては②「作業実績票」を用いて日々の実績工数を収集し、「WBS マスターシート」の進捗計上ルールにしたがって進捗率の自動計上を行う。その結果を活用して、モニタリングのプロセスにおいて変動を可視化するための③「工数・進捗管理表」や④「EVM 版実績管理表」などの各種の集計シートを生成することにより、日々の変動を可視化する。

本手法の特徴、詳細と具体的な実現方法を以下に示す。

4.3.4.　既存の EVM 適用手法との特徴の比較

始めに、本研究で提案する変動可視化手法としての「デイリーEVM 手法」の特徴を、既存の EVM の適用方法と対比して表 4-1 に示した上で、本手法独自の中核となる特徴と効果、及び実現方法を述べる。表 4-1 における「既存の EVM 手法」は、一般的に実施されている、EVM を週単位や月単位の進捗、コスト及びスケジュールの差異の分析に活用する方法を意味している。ここでは、「デイリーEVM 手法」の特徴の概要を比較項目ごとに説明し、それぞれの実現方法については後述する。なお、表 4-1 の「デイリーEVM 手法」の列に示す①から④は、図 4-3 の①～④の各シートに対応している。

表 4-1　変動把握手法の特徴

No.	実施項目	既存の EVM 手法	本手法 （デイリーEVM 手法）
1	WBS 作成	プロジェク毎に個別作成	標準的な詳細 WBS のひな型を元にした「WBS マスターシート」（①）の利用により、過去のプロジェクトの知識や実績データの活用が可能。
2	工数計画作成及び修正	工程別、機能別、WBS 別に個別作成	・「WBS マスターシート」（①）に規模や生産性等のパラメータを設定することで、WBS 別「基準工数」を一括設定 ・実績データを元に生産性等を見直し工数計画の修正を適宜実施可能
3	実績コストデータ収集	人数、期間、勤務時間、発注金額等を組み合わせて収集	実績工数の収集（②）のみによって作業コストを把握する方法であるため、データの収集や蓄積が容易
4	進捗率の計上方法とデータの収集方法	進捗率の計上は、固定比率法や成果物の量の組合せ（プロジェクト別）により、以下の各種方法を元にプロジェクト固有の方法で実施 ・進捗率報告 ・成果物の量の集計 ・ヒアリング	・WBS 別実績工数（②）や作業開始・終了情報を毎日収集し、WBS の種別に応じて設定した進捗率計上ルールにより、進捗率を自動計上（①②） ・プログラム開発工程やテスト工程では、ソースコード行数とテスト実績を補完的に利用
5	変動可視化	プロジェク毎に個別作成	出来高と実績工数から変動を可視化するためのドキュメント（③④）を自動生成

本研究で提案する「デイリーEVM 手法」の中核となる特徴と効果は以下の 3 点である。

(1) 詳細 WBS の標準化と WBS 別工数計画の一括設定

表 4-1 の No.1 に示すように、過去のプロジェクト経験を元にした標準的な詳細 WBS のひな型から成る「WBS マスターシート」を利用することにより、計画策定時の WBS

の粒度がプロジェクトマネジャーによらず標準化され、WBS の洗い出し漏れ等も防止でき、計画作成を効率良く行うことが可能となる。さらに、以下に示すように、WBS 別の計画工数をパラメータから一括で計算することや、進捗率を計上ルールに基づいて自動的に算出することが可能となる。

また、WBS を標準化することにより、原因分析や対策検討を行う際に過去のプロジェクトで蓄積されたデータや知識を活用することも可能となる。実績データを継続的に蓄積することによって、他のプロジェクトで見積りや計画策定に必要な生産性などの統計データを活用できるようになる。

(2) 基準工数の導入

工数の変動把握や進捗率の計算の基準となる、WBS 別の所用時間の基準値としての「基準工数」の考え方を導入した。この際、機能別、工程別の詳細 WBS の単位で基準工数を見積もり、設定するためには多大な労力を要するため、後述する基準工数の見積り式を用いて一括して計算できるようにしておく。具体的には、表 4-1 の No.2 のとおり、「WBS マスターシート」（図 4-3 ②）に基準工数の見積り式を組み込んでおくことで、機能別の規模や標準生産性、担当者のスキル、習熟度などの各種パラメータを設定するだけで詳細 WBS 別の基準工数を自動設定できるようにした。これにより、変動可視化のためのベースラインとなる機能別、工程別の工数計画を効率良く作成することが可能となる。また、工数計画の修正も容易に行えるため、パイロット開発や初期の開発作業等の実績データを用いて生産性等のパラメータを見直すことにより、計画の精度を高めることが可能である。

基準工数の算出根拠が明確であるため、実績データから変動の原因究明も効率良く行える。

(3) 進捗率の自動計上

表 4-1 の No.4 に示すように、事前設定した進捗計上ルールに基づき、主として実績工数の入力を元に進捗率を自動計算するしくみを構築した。この進捗率自動計上の実現方法や進捗計上ルール等については後述する。日々の作業工数や作業状況等のデータ収集（図 4-3 ②）から進捗率自動計上（図 4-3 ①）及び変動可視化用の各種ドキュメント（図 4-3 ③、④））の生成までの一連の流れを作業プロセスに組み込み、各メンバーが毎日、図 4-3 ②の「作業実績票」に実績工数を入力することで、工数や進捗率が自動的に集計されるようシステム化した。毎日出力される変動可視化用のドキュメントによって日々の変動を把握し、変動が大きい機能や人に対する重点的な監視や、変動の原因となる作業の特定等を効率良く行うことが可能となる。

以上のように、「デイリーEVM 手法」は、詳細 WBS の策定、機能別や担当者別のマネジメントのベースラインとなる基準工数計画の作成、実績工数収集、進捗率及び出来高の自動計上と変動可視化の一連のプロセスを、従来と同等以下の運用負荷で毎日実行可能とす

る手法である。これにより、精度向上と管理負荷の軽減を両立しつつ、毎日、詳細な粒度で変動を可視化することが可能となる。この可視化されたデータを活用することにより、変動の原因となる工程や人の特定、原因に応じたプロジェクトのコントロール等を効果的に行える。また、実績データを元に生産性や習熟度などの各種のパラメータを見直すことによって工数計画を必要な都度、修正することができるため、早い段階で計画の精度を高め、正確な変動の監視やコントロールを行うことを可能とする。

このように、「デイリーEVM 手法」は、3.3.1.で取り上げた Kirmani (2017) が提案している三点の工数見積りの要件（概算と実績の差異が少ない、監視や開発のプロセスに役立つ、個人レベルの詳細な工数見積り根拠の明確化）を満たしたものになっている。この考え方に加え、工数計画作成の自動化、進捗率計上の自動化、変動可視化のシステム化等によって、詳細な粒度での変動の日次での可視化を実現することによって、変動の早期把握や変動の原因分析等を効率良く行うことを可能としたものである。

次節で上記の仕組みの実現方法について具体的に説明する。

4.3.5. 変動可視化手法「デイリーEVM 手法」の実現方法

4.3.5.1. 詳細 WBS 別計画値作成の方法

初めに、詳細 WBS 別の作業時間の計画作成や実績工数の収集方法について説明する。図 4-3 の①「WBS マスターシート」を図 4-4 に示す。

この「WBS マスターシート」は、渡辺・丸山（2009）の時間単位の実績工数把握手法を参考にし、さらに工数計画の一括作成や進捗率の自動計上の仕組みを追加したものであり、本手法の中核を成すものである。

機能ID	機能名	規模	担当者	スキル	習熟度	工程係数
Axxxxxx	○○業務	800 FS	○○ ○○	1.0	1.2	1.1

工程	作業コード	作業内容	単位工数(H)	基準工数(H)	実績工数(H)	進捗率	ドキュメント進捗率	開始終了
内部設計	Axx-001	仕様理解	0.65	5.2				開始
	Axx-002	Q/A対応	0.05	0.4		30%		
	Axx-003	結合テスト仕様書作成	0.71	5.6			20%	
	Axx-004	テストデータ作成	0.64	5.1			20%	
	Axx-005	画面制御条件定義書	0.26	2.6			10%	
	Axx-006	画面項目入力チェック定義書	0.08	0.7			10%	
	Axx-007	テーブルアクセス定義書	0.26	2.1			10%	
		～ （中略） ～						
	Axx-020	ドキュメント自己レビュー	0.17	1.5		70%		
	Axx-021	ドキュメント対面レビュー(する)	0.45	3.6				
	Axx-022	ドキュメント対面レビュー(受ける)	0.37	3.0		80%		
	Axx-023	レビュー後ドキュメント修正	0.41	3.2		85%		
	Axx-024	作業方法の確認	0.00	0				
	Axx-025	手戻り(外部設計起因)	0.00	0				
	Axx-026	手戻り(方式設計起因)	0.00	0				
	Axx-027	品質チェックシート記入	0.17	1.4		95%		
	Axx-028	工程完了判断・出荷	0.06	0.5		100%		終了

図 4-4 WBS マスターシート（全体）

図 4-4 は、ある機能の内部設計工程用の WBS マスターシートの作成結果の例である。「作業内容」がワークパッケージとしての WBS に対応する。この「WBS マスターシート」はシステムの画面や帳票などの機能及び担当者の単位で工程ごとに作成する。一つの WBS の作業時間が 1 日以内の時間となるよう、数時間の粒度とし、組織内で標準化した全ての WBS を、作業内容蘭にあらかじめ記入しておく。WBS 別の「基準工数」は、各 WBS の作業時間（単位は H）の基準値であり、工数や進捗の変動を把握するための計画値（ベースライン）となるものである。WBS 別の基準工数は、WBS マスターシートのヘッダ部の各種パラメータから一括して計算できるよう、下記の計算式を組み込んでおく。

基準工数 (H)
＝ 単位工数（H／単位規模） × 規模 × スキル係数 × 習熟度係数 × 工程係数
・・・式 ①

式①において、「単位工数」は、ソフトウェアの単位規模当たりの WBS 別の平均開発工数（単位は時間（H））である。過去の類似プロジェクトでの実績データから求めた WBS 別工数比率を用い、当該プロジェクトの単位規模当りの見積り工数を各 WBS に比例配分することで、WBS 別の「単位工数」を算定する。

各工程の詳細 WBS と「単位工数」を記入したものを、機能別の WBS マスターシートを

作成するためのテンプレートとして、プロジェクト用に一つ作成する。この WBS マスターシートのテンプレートのヘッダ部に、各機能の規模、担当者のスキル係数、習熟度等を設定することで、当該機能の WBS 毎の「基準工数」が計算されるよう、式①の計算式を WBS マスターのテンプレート内に組み込んでおく。

ここで、WBS マスターシートのヘッダ部にある「スキル」は式①の「スキル係数」に対応している。「スキル係数」は、担当者別の生産性を考慮して基準工数を調整するための係数である。例えば、経験の少ない担当者の場合は、生産性が平均値より低くなる傾向にある。そこで平均的なスキルの場合を 1.0 とし、経験の少ない担当者が、同じ単位規模の機能の開発に 1.2 倍の時間を要する場合は、スキル係数を 1.2 等に設定する。このスキル係数を乗ずることにより、基準工数が平均時間の 1.2 倍に増加する。この係数については、筆者の所属組織では、組織内で実施している個人別のスキル評価結果のレベルに対応させて、標準的なスキル係数を設定する方法を採用した。実際には、個人によって生産性は異なるため、初期の計画作成時には標準的なスキル係数を採用しておき、ある工程の最初の機能が完了した時点で、個人別の実績工数から生産性を見直し、スキル係数を個人別に調整することで、その後の工数計画の精度を向上させることができる。

ヘッダ部の「習熟度」は、式①の「習熟度係数」に対応している。「習熟度係数」は、開発作業の反復を考慮した調整係数であり、過去のプロジェクトの統計値から求めた値を設定する。例えば 1 回目の作業については、作業手順に慣れていないことから標準時間に比べて 1.3 倍の時間を要するという統計データがある場合、習熟度係数を 1.3 と設定する。習熟度の具体的な設定例については、後述の工数の再見積り手法の説明の中で説明する。類似のプロジェクトの統計データが存在しないプロジェクトの場合は、過去のプロジェクトの習熟度係数を用いて計画を作成し、パイロット開発作業などの実績データを用いて見直しを行なうこととする。これらの係数を見直した場合も、WBS マスターシートのパラメータを変更するだけで基準工数を一括して修正できる点が本手法の利点の一つである。このしくみにより、パイロット開発や初期の開発作業の実績データを用いて生産性や個人別のスキル係数等を再評価し、工数計画を修正し、計画の精度を高めることができる。

なお、ヘッダ部の「工程係数」は、式①の「工程係数」に対応している。「工程係数」は、プロジェクトの特性によって、外部設計、内部設計、製造などの工程別の工数比率が異なることがあるため、過去の平均値から求められている単位工数を補正するための係数である。この工程係数については、通常は各工程の工程係数 1.0 としておき、品質が重視されるシステム開発プロジェクトなどでテスト実施密度を 1.3 倍高くする必要がある場合は、テスト工程の工程係数を 1.3 というように設定する。これによって基準工数を調整することが可能となっている。

過去のプロジェクト実績を元に、標準的な WBS から成る WBS マスターシートの標準版と WBS 別の工数比率データを組織で蓄積しておく。当該プロジェクトの生産性を反映した単位規模当りの見積り工数を、工数比率に応じて各 WBS に比例配分することにより全ての

WBS 別基準工数を一括で設定できるようにしておく。これにより、プロジェクト用の WBS マスターシートのテンプレートを容易に作成できる。プロジェクト用の WBS マスターシートのテンプレートのヘッダ部に、各機能別の規模、担当者別スキル係数、習熟度及び工程係数を設定することで、機能別、工程別、WBS 別の基準工数が一括計算される。これらの規模、スキル係数、習熟度係数等のパラメータは、「機能別パラメータ一覧」という形の一覧表として作成しておくことで、この「機能別パラメータ一覧」の情報から、プロジェクトの全ての機能の基準工数が設定された WBS マスターシートを一括生成することができる。これにより従来、手作業で行っていた WBS の策定や工数計画の作成を省力化することができる。生成された WBS 単位の基準工数を利用することでメンバー別の詳細スケジュールの策定が容易に行える。規模の変動や生産性の見直し等が必要となった場合も、「機能別パラメータ一覧」を修正するだけで、工数計画の再策定を容易に行うことができ、スケジュールの見直し等に利用可能である。

4.3.5.2. 「作業実績票」による実績把握

「WBS マスターシート」に対して、毎日の実績工数を蓄積することによって、進捗率を自動計上する仕組みを構築した。実績工数の収集は図 4-5 の「作業実績票」で行う。「作業実績票」は各担当者が入力するものである。

No.	作業コード	作業内容	開始終了	1	2	3	4	・
				月	火	水	木	
1	Axx-001	仕様理解	開始	5.5	6.5	2.0		
2	Axx-002	QA		2.5	1.5	0.4	0.5	
3	Axx-003	テスト仕様書作成	開始			5.6	7.5	
4	Axx-004	テストデータ作成						
5	Axx-005	画面制御条件定義書						
6	Axx-005	画面項目入力チェック						
7	Axx-005	テーブルアクセス定義						
・								
・								

図 4-5 作業実績票 （単位：H）

このシートには「WBS マスターシート」と対応した作業コードと作業内容を転記しておき、担当者が WBS ごとの実績時間を毎日入力する。数値の単位は時間（H）である。特定の WBS について「開始」または「終了」のフラグを記入する。「開始」フラグは省略も可能であり、作業実績時間が記入された時に当該作業の開始と見なす。

入力された WBS ごとの実績時間の累積値が、WBS マスターシートに、作業コードをキーとして毎日転記される仕組みとしている。この際、一人当たりのデータ入力時間は毎日数分であり、作業工数の 1%程度の少ない負荷で実績収集が可能である。以上の基準工数と実績工数の差異を時間単位で把握する方法が渡辺・丸山（2009）の提案手法に基づいた方法である。

上記の方法に対して本手法で拡張した進捗率や出来高の自動計上方法およびEVM を活用した変動可視化の仕組み等を以降で説明する。

4.3.5.3. 実績入力とルールによる進捗率の自動計上

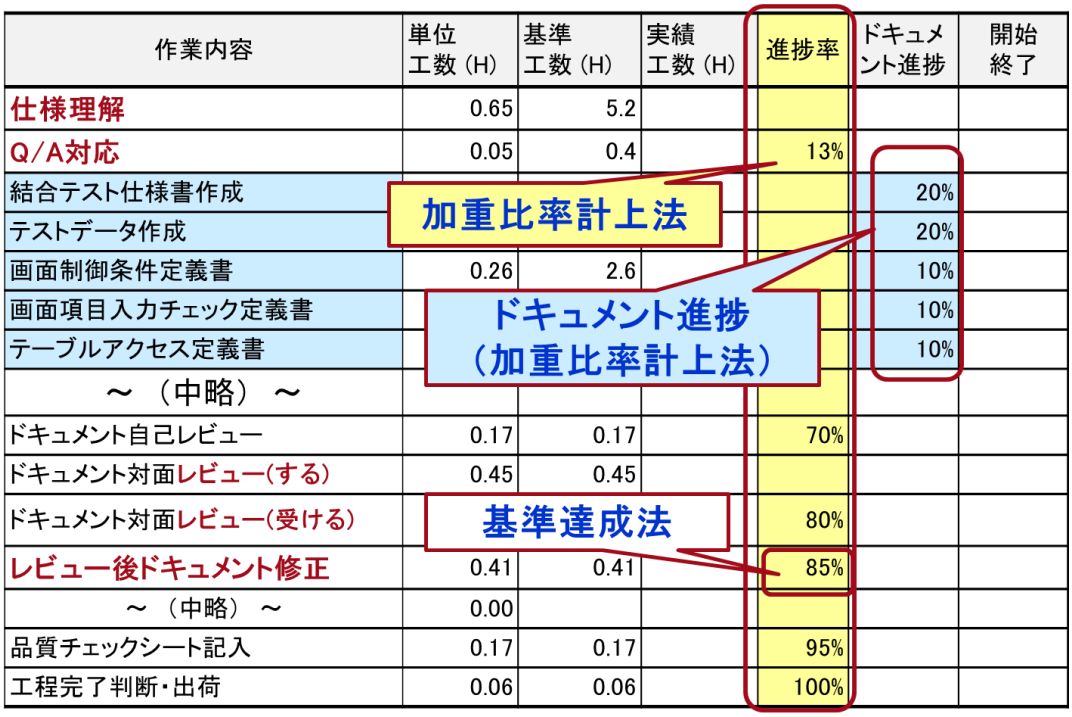
本研究では、EVM における出来高の把握の元となる、進捗率の計上自動化の仕組みを構築した。進捗の計上方法としては、作業着手時に 30%、完了時に 100%の進捗があったというように一定比率の進捗を計上する固定比率計上法や、WBS ごとに重み付けを行う加重比率計上法、複数の基準の達成をもって計上する基準達成法等がある。表 4-2 に進捗率の計上方法別に利点と欠点を示す。

これらの方法を WBS の種類に応じて適用し、WBS マスターシートと集計ツールの計上ルールとして組み込んでおくことで、進捗率を自動計上できる仕組みを構築した。以下に具体例で説明する。

表 4-2 進捗率の計上手法

No.	進捗計上手法	利点	欠点
1	固定比率計上法	少ない負荷で全体の進捗の概要を把握可能	・作業途中の進捗を把握できない
2	加重比率計上法	固定比率配分法に比べて詳細な進捗を表せる	・重みづけが経験に依存する ・作業ごとの完了の判断が主観的
3	出来高比例法	柔軟に進捗を把握可能	・進捗率が報告者の主観に左右される
4	基準達成法	達成基準を適切に設定できれば客観性の向上が可能	・達成基準の設定と判定が煩雑

図 4-6 は、WBS マスターシートによる工程別の進捗率の計上の仕組みを示したものである。時系列で順番に進めてゆく作業に関する WBS 別の進捗率は、固定比率計上法で定量化できる。例えば、作業着手時に 30%の進捗を計上し、当日中に終了すれば 100%の進捗とする。作業終了が翌日となる場合は、当日中は 30%のままであるが、1 つの WBS の基準工数は数時間であるため、ぶれ幅はその範囲に収まる。



作業内容	単位工数 (H)	基準工数 (H)	実績工数 (H)	進捗率	ドキュメント進捗	開始終了
仕様理解	0.65	5.2				
Q/A対応	0.05	0.4		13%		
結合テスト仕様書作成					20%	
テストデータ作成					20%	
画面制御条件定義書	0.26	2.6			10%	
画面項目入力チェック定義書					10%	
テーブルアクセス定義書					10%	
～（中略）～						
ドキュメント自己レビュー	0.17	0.17		70%		
ドキュメント対面レビュー(する)	0.45	0.45				
ドキュメント対面レビュー(受ける)				80%		
レビュー後ドキュメント修正	0.41	0.41		85%		
～（中略）～	0.00					
品質チェックシート記入	0.17	0.17		95%		
工程完了判断・出荷	0.06	0.06		100%		

図 4-6　進捗率の計上方法

当該 WBS の実績時間が最初に入力された時点をもって作業着手と判断する。作業終了については、前の WBS の修了が次の WBS の開始条件になっている場合、次の WBS に実績工数が記入された時点で前の WBS の終了と見なす。ただし、前後の WBS の間に依存関係がない場合は「開始」または「終了の」フラグを作業実績票に記入することとする。作業実績票には、「開始」や「終了」のフラグの記入を必要とする作業項目に印を付けておくことで、担当者が漏れなく記入できるようにしておく（図 4-5 では省略）。

各工程別の進捗率の算出には、加重比率計上法を適用する。例えば「仕様理解」と「Q&A 対応」を合わせて 13%、ドキュメント作成後の自己レビュー完了で 70%、対面レビュー完了で 80%、品質判定完了で 95%というように進捗率を割り当てる。この比率は、過去のプロジェクトの WBS 別の基準工数の工程内比率を元に算出し、標準版の WBS マスターシートに設定しておくことで、プロジェクトでの設定を効率良く行える。

以上のような進捗率や判定基準を集計ツールに組み込んでおくことにより、WBS 毎の進捗の自動計上が可能となる。

また、基準達成法の考え方も適用し、品質の確保の基準として「レビュー後ドキュメント修正の終了をもって 85%と計上する」といった達成基準に基づいて進捗率を計上する。例えば、担当者が作業実績票に工数を入力し、「レビュー後ドキュメント修正」が終了した時点で「終了」フラグを設定すると進捗率が 85%と計上される。

次に、設計ドキュメントの作成に関する WBS については、各種のドキュメトが平行して作成されることが多いため、ドキュメント作成作業の WBS 群全体に対して加重比率計上法

の考え方を適用する。WBS マスターシートの「ドキュメント進捗率」の欄に、当該工程のドキュメント作成作 WBS 群全体を 100%とした時のドキュメント別の進捗比率を設定しておく。例えばドキュメント作成作業の中における比率を「画面項目定義」で 10%、「テーブルアクセス定義」で 10%、「テスト仕様書作成」で 20%というように割り当てる。ドキュメント作成作業全体の比率が内部設計工程の 30%の場合、ドキュメント種類別の工程内進捗率としては、「ドキュメント別進捗率×0.3」が計上される。さらに、1 つのドキュメントの作業途中の進捗については、着手した時に 10%とし、50%に到達するまでは基準工数に対する実績工数の比率を計上する。それ以降は自己レビュー終了時点で 70%というように、加重比率基準法を応用した進捗計上ルールによって、設計ドキュメント作成途中の進捗率を把握可能とした。

以上のように進捗計上ルールを設定しておくことで、各担当者が毎日の作業実績工数を入力するだけで進捗率が自動計上され、出来高（基準工数 × 進捗率）と実績工数の比較により生産性の変動把握や EVM によるコスト予測が日次のサイクルで可能となる点が、従来の手法にはなかった特徴である。これは一つの WBS を数時間程度の詳細な粒度とし、各 WBS に進捗率計上ルールに従った進捗率の計算ロジックを集計のためのシステムに組み込んでおくことで実現可能となったものである。

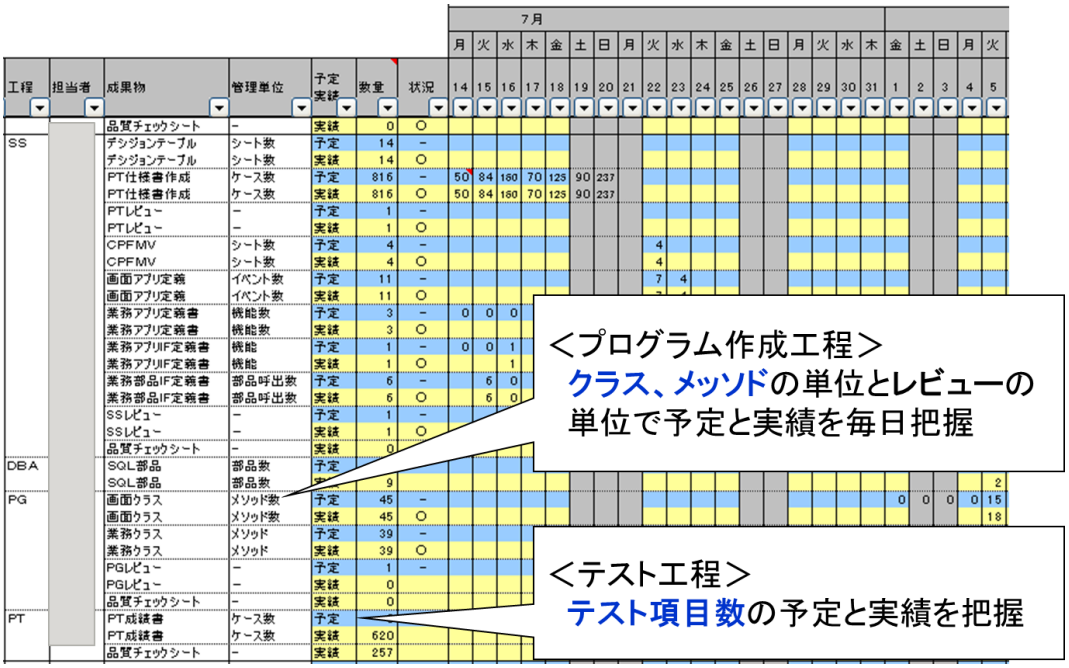


図 4-7　プログラム作成工程における実績管理シート

プログラム開発工程においても、実装単位が数時間～2 日程度のなるべく短い単位で一つの WBS とすることで、上記と同じ方法が適用できる。ただし、一つの開発単位の規模が大きく作業期間が数日に渡る場合は、加重比率計上法や基準達成法に加え、メソッド数等の

計画値と実績値からの進捗率計上も併用する。図 4-7 に、プログラム開発工程における実績管理シートの例を示す。詳細は省略するが、開発単位の規模が大きい特定のプログラムについては、図 4-7 によって求められる進捗率を WBS マスターシートに転記するようにしておく。また、テスト工程におけるテスト実施作業は、毎日のテスト項目数の計画値と実績値を進捗率の算出に用いる。

上記の進捗計上ルールを組み込んだ WBS マスターシートのテンプレートをプロジェクトで一つ準備し、機能数分複写して各係数を設定するだけで、全ての機能の工数計画作成と進捗率の計上等の準備が完了する。この準備の後、担当者が毎日作業実績票に実績工数を入力することにより、日々の機能別、工程別の進捗率が自動計上され、出来高（基準工数 × 進捗率）や実績工数と合わせて集計・加工することで、図 4-3 における③や④などの変動を可視化するための各種ドキュメントが毎日自動生成される。このように実績工数等のデータから進捗率等の情報が自動集計されるため、報告に基づく進捗等の把方法における主観性や曖昧性の問題を排除できる。

以上の変動可視化手法（デイリーEVM 手法）のプロセスの流れを、図 4-3 を元に再整理しておく。機能別に展開された「WBS マスターシート」の WBS が転記された「作業実績票」に対して、担当者が WBS 別実績工数と開始や終了のステータスを毎日記入する。この実績値が「WBS マスターシート」に転記され、累積工数や進捗率が毎日自動計算される。この情報を元に各種の一覧表を作成することで変動状況を可視化する。このように、毎日の実績データから進捗率や工数、出来高を自動集計し、日次で変動を可視化するためのドキュメントの出力までをシステム化したものが、本研究で提案する変動可視化手法（デイリーEVM 手法）である。

4.3.5.4. 工数・進捗集計表による変動の可視化

出力される変動可視化用の各種ドキュメントによって、様々な観点で日々の変動を確認することが可能となる。以下に、変動を可視化するための各ドキュメントについて説明する。

基本情報			内部設計工程						
システム情報			担当者	作業時間			生産性（内部設計）	進捗率	ドキュメント進捗率
機能ID	機能名	規模		基準作業時間	実績作業時間	基準作業時間消化率			
合計		28633		2361.4	3274.3				
平均		1145.3		94.5	131.0	139%	0.0	90%	92%
ABCAA03	機能A	1512	A	140.2	268.9	192%	17.8	100%	100%
ABCAA04	機能B	2723	B	230.7	339.0	147%	12.4	100%	100%
ABCAA05	機能C	1055	C	102.1	121.2	119%	11.5	96%	85%
ABCAA06	機能D	461	D	47.0	0.3	1%	0.1	20%	0%
ABCAB03	機能E	1235	E	124.1	155.3	125%	12.6	90%	100%
ABCAB05	機能F	1677	F	129.6	214.1	165%	12.8	86%	88%
ABCAB07	機能G	1250	G	125.6	180.9	144%	14.5	79%	80%
ABCAB08	機能H	1783	H	151.4	153.6	101%	8.6	90%	100%
ABCAB09	機能I	1839	I	230.2	162.8	71%	8.9	85%	68%
ABCAB12	機能J	1033	J	61.9	60.2	97%	5.8	100%	100%

図 4-8　工数・進捗集計表の例

具体的な変動の可視化情報の活用方法については、次章で事例を元に例示する。

図 4-8 は「工数・進捗集計表」の例である。機能別の工数と進捗に関する計画と実績の差異を毎日集計して出力するものであり、機能別の「基準作業時間消化率」（基準工数に対する実績工数の比率）や進捗率が一覧表示される。基準工数と実績工数の比率が、事前に設定したしきい値よりも増加した場合に背景色等が変わるようにしておくことで、変動の大きい機能や担当者を視覚的に識別することができる。図 4-8 で数値欄の背景が網かけ状態に変わっている箇所は、実績値がしきい値を超過したことを示している。このような変動状況を各種の観点での分析し、可視化するためのドキュメントを毎日、自動的に出力することによって、運用の負荷を増やすことなく、重点的な監視が可能となる。

4.3.5.5.　EVM の拡張による日次での分析やコスト予測

「WBS マスターシート」によって集計される工数や進捗率を利用して EVM (Earned Value Management) の考え方を応用することにより、生産性の評価やスケジュール及びコスト予測等を毎日確認することが可能となる。従来の EVM の適用方法は、サブシステムやチーム単位で、週や月を単位とした一定期間のスケジュールやコストの傾向を評価するという使い方に止まっていた。本研究で提案するデイリーEVM 手法は、人や機能の単位で、生産性やコスト予測等を毎日実施できる点が特徴である。

図 4-9 に示すように、基準工数を PV (Planed Value)、基準工数×進捗率を EV (Earned Value)、工数実績を AC (Actual Cost)として EVM を適用する。

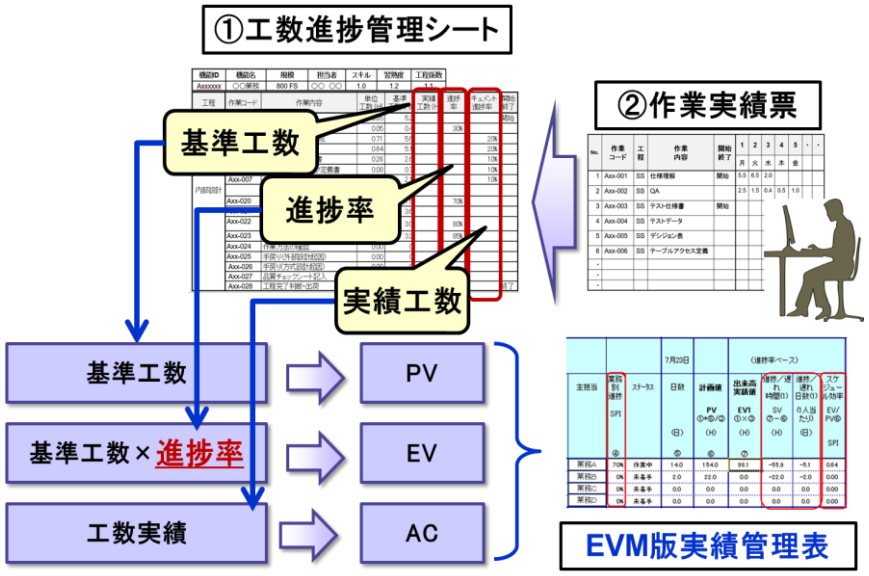


図 4-9　EMV の適用方法

この考え方を適用することで進捗とコストを統合的に把握するための EVM 版実績管理表」を図 4-10 に示す。

機能別情報			進捗		EVM計画・実績				コスト		コスト指標		
No.	機能	工程	基準工数 (H) (1)	進捗率 (%) (2)	PV (H) (3)	EV (H) (4)	SV (H) (5)	差分日数 (Day) (6)	AC (H) (7)	CV (H) (8)	CPI (9)	EAC (H) (10)	VAC (H) (11)
1	画面A	SS/PT Spec.	140.2	85%	130.2	119.1	−11.0	−1.0	228.1	109.0	0.52	268.4	128.2
		DBA	31.2	50%	10.4	15.6	5.2	0.5	12.5	−3.1	1.25	25.0	−6.2
		PG	71.2	0%	0.0	0.0	0.0	0.0	0.0	0.0	0.00	71.2	0.0
		PT	75.9	0%	0.0	0.0	0.0	0.0	0.0	0.0	0.00	75.9	0.0
2	画面B	SS / PT Spec	230.7	90%	230.7	207.7	−23.1	−1.2	299.6	91.9	0.69	332.9	102.2
		SS		—	—	—	—	—	—	—	—	—	—
		DBA	19.5	40%	9.8	7.8	−2.0	−0.2	8.6	0.8	0.90	21.6	2.1
		PG Main	106.6	0%	0.0	0.0	0.0	0.0	0.0	0.0	0.00	106.6	0.0
		PG Sub		—	—	—	—	—	—	—	—	—	—
		PT	147.7	0%	0.0	0.0	0.0	0.0	0.0	0.0	0.00	147.7	0.0
		PT		—	—	—	—	—	—	—	—	—	—

図 4-10　EVM 版実績管理表

図 4-10 に含まれる主要な管理項目や指標について以下に説明する。下記の括弧付き番号は、図 4-10 の各項目の番号に対応している。

(1) 基準工数（H）: WBS 別の基準工数
(2) 進捗率（%）: WBS マスターシートを用いて算出された進捗率
(3) PV（H） ＝ 基準工数（H） × 経過日数／予定日数
(4) EV（H） ＝ 基準工数（H） × 進捗率（%）
(5) SV (Schedule Variance) (H) ＝ EV (H) － PV (H)
(6) 差分日数（day） ＝ SV (H)／保有工数（H／day）
(7) AC (Actual Cost)（H）＝実績工数（H）
(8) CV (Cost Variance)（H） ＝ EV (H) － AC (H)
(9) CPI (Cost Performance Index) ＝ EV／AC
(10) 予想総コスト EAC (Estimate At Completion) ＝ AC ＋ (BAC －EV) ／CPI
(11) 完了時コスト差異 VAC (Variance At Completion) ＝ EAC－BAC
（BAC: Budget At Completion）

「EVM 版実績管理表」上には、機能別、工程別に、上記に示す(1)から(11)の各種指標の変動が、毎日表示される。プロジェクト全体やチームを管理単位として月や週のサイクルで俯瞰的な評価を行う EVM の活用手法と比べて、本研究で提案する「デイリーEVM 手法」では、このように、機能別、担当者別、工程別の変動を、EVM の各種指標を利用して毎日可視化することができる。例えば、差分日数（6）によって、機能別の進捗が何日遅れかを、毎日把握でき、遅れを抑制するための対策等を、機能や人別の状況に合わせて検討することに活用できる。また、(10)の完了時予想総コスト（EAC）が日々更新されるため、現在の状況でプロジェクトが推移した場合の最終的なコストやスケジュールの見通しを、ステークホルダの間で共有し、対策の検討に活用することが可能となる。

4.3.6 進捗・コストの変動可視化手法のまとめ

本節では、進捗とコスト（工数）の日々の変動を、機能別、担当者別に、詳細 WBS の単位で可視化するための手法を提案した。一つの WBS の工数が数時間から 1 日以内の詳細な粒度の WBS 別に、実績データ入力から進捗率を自動計上する仕組み等により、運用負荷を増やすことなく詳細な粒度の進捗や工数の変動を可視化することが可能となった。この際、標準的な WBS に対応した単位工数を用いることでベースラインとしての WBS 別工数計画（基準工数）の作成を一括で実施可能とし、この基準工数と進捗計上ルールを元に、実績データから進捗率を自動計上するようにシステム化することによって、計画作成や実績収集の運用負荷を軽減し、精度の向上と運用負荷の削減を両立させた。

この変動可視化手法のインプットとなる重要なメトリクスとしてのソフトウェア規模の変動の把握方法については、次節で提案する。

4.4. ソフトウェア規模の変動可視化手法

本節では、前節での進捗や工数の変動可視化手法におけるインプットとしての情報の一つである、ソフトウェア規模の変動を可視化する手法を提案する。本手法は、前節の進捗及び工数の変動可視化手法であるデイリーEVM 手法に包含される位置付けにある。

4.4.1. ソフトウェア規模の重要性と課題

システム開発プロジェクトのマネジメントにおいて、開発対象ソフトウェアの規模は、開発工数の見積りやプロジェクト計画の策定、品質管理等の定量的管理の基礎となる重要な尺度である（図 4-11）。

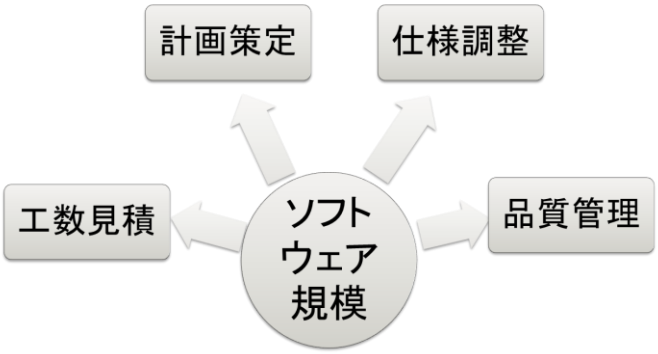


図 4-11　ソフトウェア規模を元にした主要なプロジェクト管理要素

しかし、ソフトウェア開発プロジェクトは、初期段階では要件の曖昧性が大きいことに加え、設計や開発工程の中で発生する仕様の変更等によって、プロジェクトの進展に伴い、ソフトウェア規模の変動や増加が発生することが多い。ICT の活用領域の広がりに伴い、要件の曖昧性が大きくなる傾向にあり、仕様変更等によりソフトウェア規模の変動が発生することが多くなっている。

JUAS (2016) の調査結果によると、例えば画面数は計画値に対して最終的に平均値で 20%増加している。**図 4-12** に、社団法人　日本情報システム・Z 協会、『2011 年版調査報告書ユーザー企業ソフトウェアメトリックス調査』によるグラフを示す。これは、システムの画面の数が計画値に比べて変動した件数のグラフであり、横軸は計画時の画面数に対する実績値としての画面数の比率を示しており、縦軸は画面数を表している。このグラフより、計画時よりも画面数が増加しているプロジェクトのほうが明らかに多くなっていることがわかる。このグラフはプロジェクトの最終的な結果を表しており、途中で仕様追加等が発生した場合に規模を縮小させるための調整などを行った結果であることを考慮すると、プロジェクトの進行中においてはこれ以上の規模の増加が発生していると推測される。このような状況において、規模増加等の状況を正しく把握できないままプロジェクトが進むと、

様々な問題が発生する。しかし規模の見積りや計測を何度も行うと負荷が高くなるため、規模の変動状況を効率良く把握しマネジメントに活用する効果的な手法はなかった。本章では、ソフトウェア規模の変動を継続的に可視化することで、早い段階で仕様の調整や要員追加のための再見積りを可能にし、後工程での問題発生を防止するための手法を提案する。

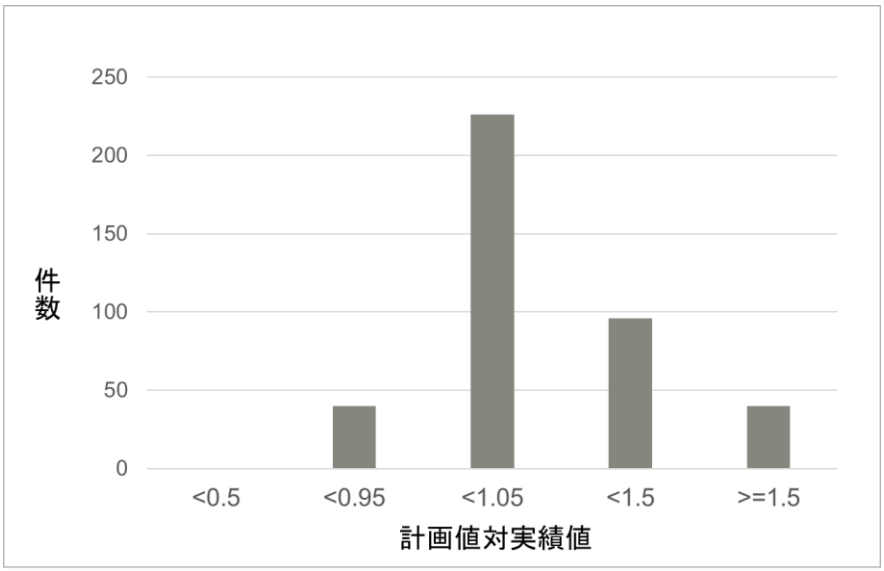


図 4-12　画面数の計画値対実績値の度数分布

（社団法人　日本情報システム・Z 協会、「2011 年版調査報告書　ユーザー企業ソフトウェアメトリックス調査、図表 6-14 より」

4.4.2.　ソフトウェア規模の変動量把握の必要性と課題

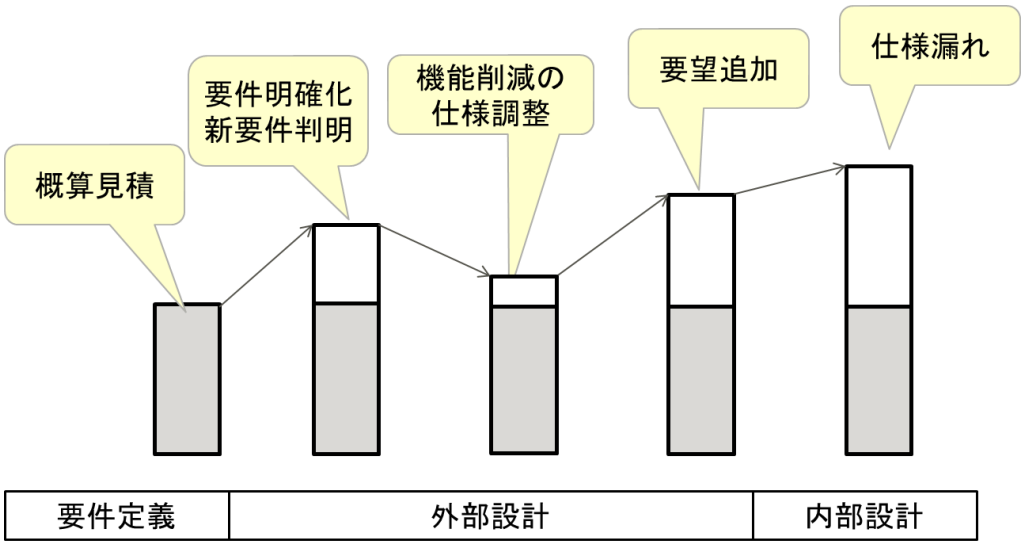


図 4-13　ソフトウェア規模の変動と対処の例

図 4-13 に、開発プロジェクトにおけるソフトウェア規模の変動の例を示す。

設計工程の進展に伴い、ソフトウェア規模の増加が判明した場合は、開発コストを予算内に収めるため、機能削減等の仕様に関する調整が必要となる。要件変更や仕様追加が避けられない場合は、要員の増員やプロジェクト計画の修正が必要となる。しかし規模の変動量を設計工程で精度良く把握することは、後に述べる理由により実行が困難なのが現実である。

規模が増加し、その影響がプログラム開発工程以降で工数増加や進捗の遅延となって顕在化してからでは、仕様の調整は難しくなる。要員追加等を行う場合も、このような状況になってからでは対策が後手に回り、コスト増加や品質悪化を招くことが多い。したがって、ソフトウェア規模の変動状況を、設計工程の段階から継続的に把握できるようにすることは、重要な課題である。

ソフトウェアの規模を表す方法としては、SLOC (Software Lines of Code) 、ファンクションポイント（Function Point）法（以下、FP 法と略す)、機能数（画面数、帳票数）などがある。第 3 章の表 3-6　LOC と FP 法及び FS 法の強みと弱みの比較」に示すように、SLOC での規模見積は、有識者の経験や暗黙知への依存性が高く、仕様と規模の対応関係もわかりにくいため、設計段階で規模の変動を客観的に繰り返し把握することは困難である。FP 法は、計測手法が標準化されており、計測者によるぶれは少ない。しかし FP 法は外部仕様だけでなくデータ構造等の把握が必要であり、概算見積のための NESMA (2005) の「FP 概算法」や「FP 試算法」などもあるが、設計工程の中で、繰り返し FP 法で規模を算出するには作業負荷が高い。また、画面数や帳票数等の機能数では、画面単位での追加・変更による影響はわかっても、画面内の機能追加などは表せないため、規模変動の把握には適さない。

以上の理由により、SLOC、FP 法、機能数のいずれを用いた場合でも、設計途中での規模の見直しは、作業負荷が高いために行われないか、設計者の経験や勘に頼った不正確な情報にとどまることが多いと言える。

4.4.3. 規模変動の可視化手法

4.4.3.1. 規模変動可視化のための規模測定方法の選択

ソフトウェア規模の変動量把握の課題に対する解決手段として、設計段階から継続的に規模の変動を可視化するための手法を提案する。

最初に、継続的に規模の変動を可視化するために適している規模計測方法を選択する。以下に、規模の変動を、継続的に可視化するための要件を整理する。

(1) 理解容易性

外部設計段階から規模を計測可能であり、画面や帳票等のレイアウトなどの、利用者やプロジェクトオーナーが理解容易な情報との対応関係が明確であること

(2) 客観性、精度

人に依存せずにぶれのない規模を算出可能であること

(3) 測定の容易さ

計測や再計測に必要な時間が短時間であること

規模変動の継続的可視化に適した規模測定法を選定するため、第 3 章の表 3-6　LOC と FP 法及び FS 法の強みと弱みの比較」を元に、上記の(1)～(3)の要件への適否を比較の観点とした比較結果を、表 4-3 に示す。

表 4-3　規模変動可視化の観点でのソフトウェア規模見積り手法の比較

No.	測定方法	(1) 理解容易性	(2) 客観性精度	(3) 測定の容易さ	規模変動可視化への適合性
1	SLOC（ソースコード行数）	× 根拠が曖昧	× 担当者の経験等に依存	△ 業務経験が必要	×
2	機能数（画面・帳票数）	○	× 粒度が粗い	○	△ 正確に把握できない
3	FP (Function Point) 法	△	○	△ 計測の負荷が大きい	△
4	FS (Function Scale) 法	◎ 外部設計書と対応	○ 計測者によらず同一	○	○ 差分計測が可能

理解容易性については、FS 法は、画面レイアウトや帳票レイアウトなどの、利用者にわかりやすい設計情報を元に規模を計測可能なため、ソフトウェアの仕様と規模の対応関係について、利用者などのステークホルダが理解容易である。客観性と精度については、FS 法では画面などの構成要素を元に機械的に規模を算出するため、測定者に依存せずにぶれのない機能規模を算出することができる。測定の容易さについて、FS 法は、画面ごとのコントロールの数をワークシートに記入するだけで、画面単位やサブシステム全体の規模等を自動的に計算できるようになっている。測定に要する時間に関しては、測定方法を理解すれば一画面当たり数分で算出できる。また、「画面項目定義書」等の定型化された画面の外部仕様書等を適用することにより、FS 値を自動的に計算するための計測ツールを作成す

ることも容易であり、規模計測の自動化が可能となる。さらに、一度、規模を計測しておけば、仕様変更等があった場合には、該当の画面やコントロールの数だけを部分的に見直してワークシートに記入することで、最新の規模を求めることも容易である。このため、差分計測が可能となり、規模の変動を継続的に集計することに適している。

上記の比較検討結果により、FS 法が、外部設計情報から、ぶれのない機能規模を少ない負担で繰り返し測定できるため、規模の変動状況を継続的に把握する上で適していると判断した。ただし、規模変動可視化に関する(1)～(3)の要件を満たす規模計測方法であれば、他の方法を採用することも可能である。

4.4.3.2. 規模変動の可視化手法

以下に、上記の FS 法を適用した規模変動の可視化手法のプロセスを説明する。

(1) 規模計測の集計自動化

継続的に規模計測を行うため、作業プロセスに FS 計測を組み込んでおく。設計担当者は、担当画面の規模を定期的又は規模に影響する変化があった場合に再測定する。FS 法では、「FS 集計ワークシート」に、画面レイアウト設計書を元にしたコントロールの種類別個数を記入することで、画面や帳票などの機能別の規模を自動集計する。設計担当者が入力した FS 集計ワークシートをサーバにアップロードし、システムやサブシステム単位の規模一覧と規模分布図を自動出力し、一連の規模可視化プロセスを自動化することで、継続的に規模変動を可視化する。

(2) 規模変動可視化のタイミングの設定

規模の変動を継続的に可視化し、ステークホルダとの間で共有するためには、規模計測の結果を集計し、可視化するタイミングをプロジェクト計画として設定しておくことが重要である。計測及び集計のタイミングを決めておくことで、設計担当者の規模計測を同期させ、プロジェクトオーナーとの間での仕様やスケジュールの調整などを、時期を逃さずに適切なタイミングで行うことが可能となる。

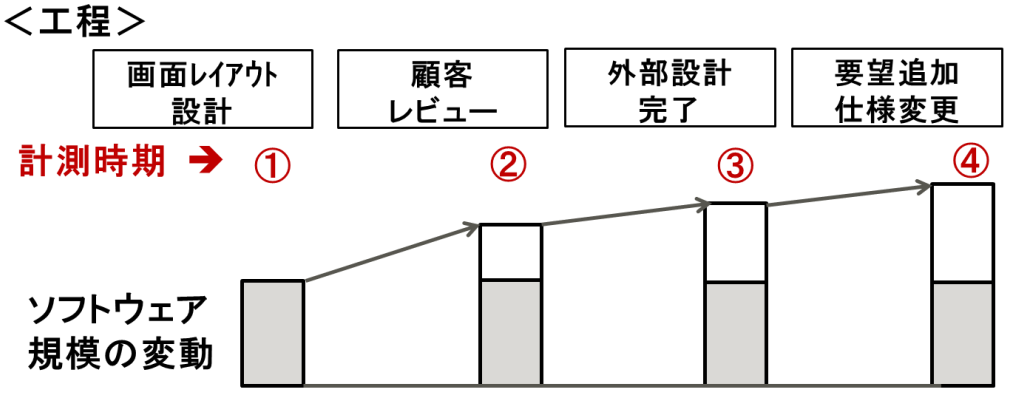


図 4-14　規模変動の計測時期

具体的な規模計測のタイミングの一例を、**図 4-14** に示す。図に示す規模計測時期について以下に説明する。①から④の数字は**図 4-14** の計測時期（①~④）に対応している。

① 最初に FS の詳細計測が可能になる外部設計工程において、画面レイアウトと主要機能（検索、表示、更新、登録）の設計を終える時点

② 顧客レビュー終了時点（レビュー指摘を設計に反映させた時点）

③ 外部設計終了時点

④ 上記の時点以外に規模への影響が大きい要望や変更が必要になった際に追加で測定する

(3) ステークホルダとの事前合意事項

各時点での規模の変動量に応じて仕様やコストの見直しを行うことについては、プロジェクトオーナーを含むステークホルダとの間で事前に合意しておくことが重要である。例えば、SI ベンダーが開発を受託する形態のプロジェクトにおいて、受託者がこのような事前説明や合意形成を行わず、規模増加が発生してから機能削減や費用増加の交渉を行った場合、受託者側の見積責任などを問われ、調整が難航するリスクがある。これに対して、仕様やコストの見直しについて事前に理解を得ておくことにより、仕様調整や計画変更の際に関係者の協力が得られやすくなる。

(4) 規模分布の可視化

規模の変動は、個々の機能別の規模の増減だけでなく、後述するように、規模の大小やシステム全体での機能別の分布状況が、習熟度との関係によって生産性や要員の割り当てに影響を及ぼす。このため、規模計測のタイミングに合わせて、規模分布状況を表すグラフを、毎回自動的に出力することで、分布状況の変動がわかるようにした。

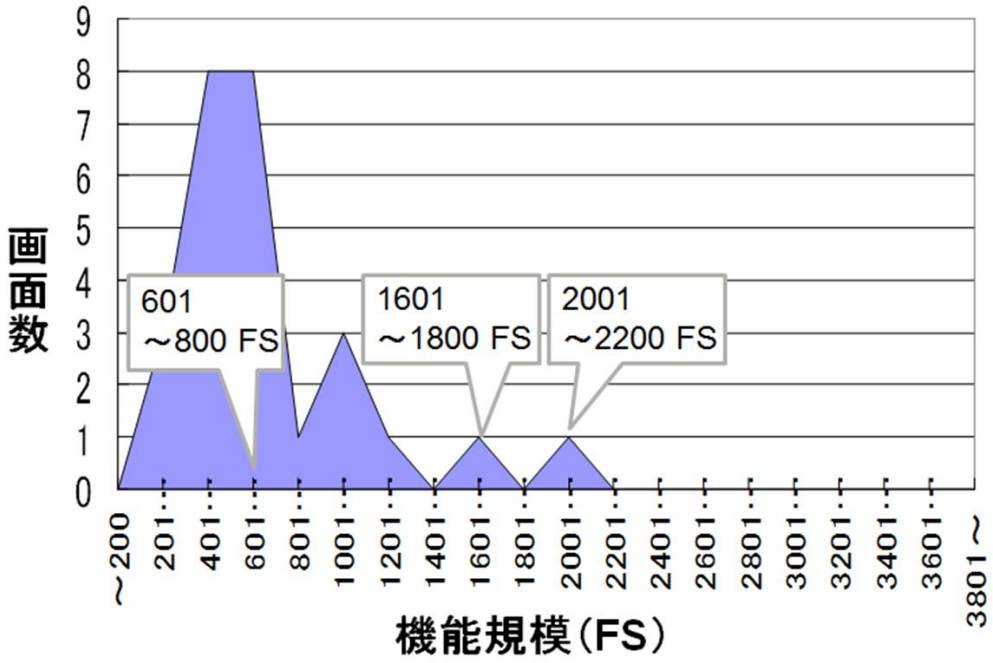


図 4-15　画面単位の規模分布図

図 4-15 は、FS 法による画面ごとの規模計測結果を元に、画面単位の規模の分布状況を表したものである。横軸に 200 FS ごとの FS 値、縦軸に画面数による度数分布で表現している。この機能分布図により、どの程度の規模の画面が何画面存在しているかを俯瞰でき、工数見積や要員配置、スケジュール策定等に活用できる。たとえば、規模が大きい画面は難易度も高いことが多いため、経験が豊富でスキルの高い要員に割り当て、経験の少ない要員には規模が小さめの画面を割り当てて複数画面を繰り返し担当させ、習熟効果による生産性向上を図るというように、適切な要員の割り当て等に活用できる。

規模の計測の都度、規模分布図を更新し、規模の変動状況を分布状況として可視化することにより、プロジェクトマネジャーが、規模変動の状況をシステム全体で俯瞰的、包括的に把握でき、対策を打つべきポイントを把握しやすくなる。

4.4.4.　デイリーEVM 手法における規模変動可視化手法の位置付け

規模変動の可視化手法は、進捗及び工数を中心とした可視化手法である「デイリーEVM 手法」のマネジメントプロセスに組み込んで利用することによって、プロジェクトの変動を総合的に可視化することが可能となる。この規模、進捗、コストの可視化手法と、次章で提案する変動に対する再見積り手法を含むコントロール手法と一体として、変動マネジメント手法を構成するものである。これらを合わせたプロジェクトへの適用による評価については、次章で示す。

なお、本研究におけるデイリーEVM 手法では、外部設計情報から規模を計測でき、計測

者によって変わることなくぶれのない数値が得られるという特長を有するFS法を利用する方法を提案したが、本手法は、特定の規模計測手法に依存するものではなく、外部設計情報から規模を計測可能な手法であれば、他の計測方法の利用も可能である。

4.5. プロジェクトの変動に対するコントロール手法

4.5.1. 変動に対するコントロールの知識の形式知化

前節までで、プロジェクトの各種の変動を、詳細な管理粒度で運用負荷を増やさずに可視化するための手法について提案した。本節では、可視化された変動情報を活用しプロジェクトを適切にコントロールするための過去のノウハウを形式知化することによって知識の活用や継承を行う方法について提案する。図 4-2 の知識関連図に示すように、プロジェクトマネジャーは変動に対して様々な経験に基づく知識や情報を元にプロジェクトのコントロールを行う。

本研究の対象とするプロジェクトは、主に、企業や組織内で利用するシステムの開発プロジェクトを想定しており、要求を満たす品質のシステムを、一定の予算の中で目標とする期間内に開発するためのマネジメントを要求されるプロジェクトである。この場合、品質、コスト、納期のうちの優先順位としては、納期の優先度が最も高いことが多い。このため、スケジュールの遅延が発生した場合の対策としては、スケジュールの遅れを取り戻すために要員やコストを増加させるというコントロールを行うことが多い。したがって、以下に示すプロジェクトのコントロール手法の提案や、適用事例による手法の評価においては、スケジュールや工数の変動が発生した場合に、要員を追加投入する場合の工数の再見積り手法に焦点を当てて述べる。ただし、納期よりも要求や仕様を含む品質を重視するプロジェクトにおいても、スケジュールやコストを見積もるためには、規模やスケジュールを考慮した工数の再見積りが必要となるため、本研究で提案する再見積り手法を活用することが可能である。

企業内のシステム開発プロジェクトで、進捗や工数の変動が発生した場合には、規模や生産性などの各種の管理要素のどの要素に関連した原因かを分析し、原因に応じた対処を行なうことが必要となる。例えば、規模の増加が判明した場合に、納期やコストを予定通りにコントロールするためには、プロジェクトオーナーとの間で、機能の削減のための調整を行うことが必要となる。この際には、プロジェクトマネジャーは、規模の増加によってコストやスケジュールにどのような影響が発生するかを定量的に説明できる必要がある。また、スケジュールの遅延の原因が、生産性や設計品質等に起因したものである場合は、作業工数の増加や要員追加等の対策を講ずる。この際の必要要員数の見積りにおいては、要員追加に伴う生産性低下などの影響を考慮して追加要員数を見積もるが、このような見積りは、有識者の経験や勘に依存していることが多いと考えられる。

本節では、前節までの変動可視化手法（デイリーEVM 手法）によって得られる各種のデータを用いた分析に基づいて、コントロールを行う際の知識を形式知化する方法を提案する。スケジュールの見直しや要員追加等を行う場合の再見積りについては、プロジェクト

マネジャーの暗黙知となっている見積り方法を、再見積方式として形式知化する。具体的には、習熟度や開発期間などの、生産性やコストへの影響度合いを考慮した工数見積式を定式化することにより、変動や計画変更に対するコストや要員数を的確に算出する見積り方式を提案する。さらに、再見積方式による計画変更方法を含めて、変動の原因別のコントロール方法をパターン化し、デイリーEVM 手法と合わせた変動マネジメント手法としての全体像を提示する。

本手法のねらいは、変動に対する可視化手法と、コントロールに必要な知識を形式知化したコントロール手法によって過去のプロジェクトマネジメントの知識を活用し、仕様調整や要員追加等の変更等を早めに行うなどの、プロアクティブなマネジメントによるプロジェクトの円滑な推進を可能とすることである。

4.5.2. 計画と実績の差異発生の要因と課題

ソフトウェア開発プロジェクトにおいては、スケジュールの遅延が明らかになってから要員を投入しても、Brooks (1995) による“ブルックスの法則”として知られているように、さらなる要員の追加投入や品質悪化などの悪循環に陥るリスクがある。このような問題の発生を未然に防止するためには、規模変動や進捗や工数の変動の正確かつ短いサイクルでの把握に加え、遅れの傾向や原因を明確にし、問題が拡大する前に、追加要員数などの正確な見積りに基づいた早い段階での対応が重要となる。

変動の原因に応じて、正確な再見積りを行うためには、以下のような課題がある。

(1) 規模と開発期間の関係により作業の反復回数が変わると生産性に影響が生ずるが、その影響度を定量的に見込む手法はなかった。
(2) 工数や要員数を増加規模と生産性から単純計算するだけでは、要員の増加等に伴うオーバーヘッドなどの考慮が漏れる可能性がある。
(3) スケジュール確保を優先し要員追加や作業分割を行う場合は各種のオーバーヘッドが生ずるが、その見積り手法は確立されていない。

上記(1)から(3)の課題によって、見積りが過少となると、オーバーヘッドなどの影響によって要員投入の効果が得られなくなることがある。このような過少見積りを防止するための対策として、規模増加や開発期間の変動を考慮し、実績データに基づいて増加工数を算出するための見積方式と、その結果を活用したマネジメント手法を提案する。以下にそのねらいと具体的な方法を述べる。

4.5.3. 変動要因を考慮した再見積り手法

4.5.3.1. 再見積方式の概要

生産性や工数は様々な要因によって変動するが、各変動要因が生産性や工数にどの程度影響を及ぼすかを考慮した工数見積や管理のための手法は確立されていない。そこで、あらかじめ規模や工数、期間などの変動が発生するケースを想定し、各種変動要因と生産性や工数との間の因果関係を明らかにすることで、工数や要員数の再見積りを行うための計算式を含む再見積り方式を体系化する。この方式の狙いは、各種のオーバーヘッドを考慮した工数の再見積りによって、過少見積りとならない適切な工数見積りを行い、コスト削減のための早めの仕様調整、追加要員の投入計画策定、追加要員に対する教育やサポート体制の準備等の予防的なコントロールを可能とすることである。これにより、経験豊富なプロジェクトマネジャーの再見積りとコントロールのノウハウを形式知化するものである。見積式には、工数に影響を及ぼす要因のうち、従来はプロジェクトマネジャーの経験で見積もっていた、以下のような、プロジェクトの変動に伴う影響要因や観点を組み込む。

(1) 作業の反復回数による生産性への影響

過去のプロジェクト実績から、類似作業の反復回数と生産性の関係を定量化する。

(2) 要員増加に伴うオーバーヘッド

要員が増加すると仕様引き継ぎやレビューワの工数等が増加する。この影響を過去のプロジェクトの分析から算出する。

この見積式により、実績データを元に規模増加や開発期間の変動に対する増加コストや追加要員数を算出し、適時に仕様調整や精度の高い要員追加等の対応を行えるようになる。

図 4-16 に、変動要因を考慮した工数の再見積方式の全体像を示す。

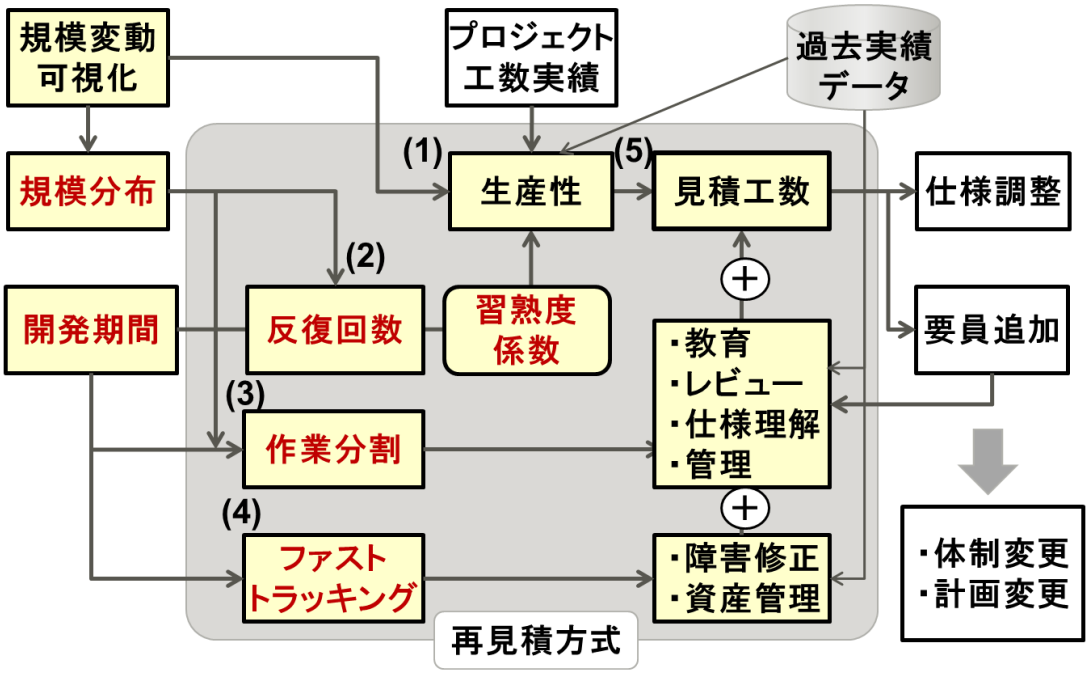


図 4-16　変動要因を考慮した工数の再見積方式

図 4-16 に示す見積り手順の概要を示す。以下の括弧付き番号(1)～(5)は、**図 4-16** の中の括弧付き番号に対応している。

(1) 規模、開発期間、実績工数等からの生産性の見積り

(2) 規模分布等からの一人当たりの作業の反復回数の算出

(3) 作業分割を行う場合の工数加算

(4) ファストトラッキングを行う場合の工数への影響の考慮

(5) 機能別のソフトウェア規模に、(1)で求めた生産性と、(2)から求めた習熟度を乗じ、(3)と(4)で考慮した工数を加算することによって、工数を算出する。

以下に、上記手順の(2)、(3)、(4)、(5)の手順における見積りの考え方を説明する。

4.5.3.2.　反復回数と生産性の関係の考慮

類似の作業を反復して実施すると、作業に習熟するにつれて生産性が向上する。この反復回数と生産性の間の定量的な関係を知ることができれば、作業の反復回数を考慮した精度の高い工数見積が可能となると考えられる。

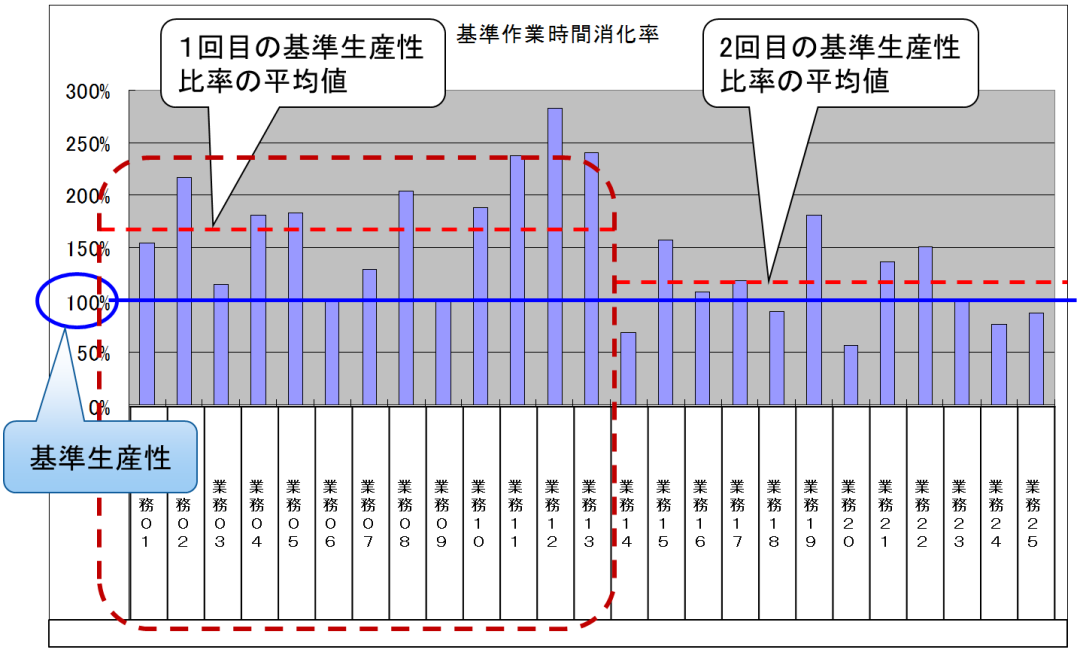


図 4-17　反復回数と生産性の関係の一例

図 4-17 は、あるウォーターフォール型での企業内システム開発プロジェクトにおける内部設計工程からプログラムテスト工程までの、機能別の生産性をグラフ表示したものである。当該プロジェクトでは、基本として一人の担当者が複数の業務機能の開発を担当したが、一部の担当者は、大きい規模の業務機能を 1 機能だけ担当したケースもある。横軸の業務 01～業務 25 は機能名であり、縦軸は基準となる生産性を 100%とした場合の相対的な産性の比率（%）を棒グラフで表示している。ここで、基準となる生産性は、変動可視化手法の中で利用している単位規模当たりの「基準工数（H)」を基準として採用した。グラフの左側の、点線で囲んだ業務 01 から業務 14 までが、それぞれの担当者が 1 回目に開発した機能であり、その右側の業務 15～業務 25 は、各担当者が 2 回目に開発した機能である。1 回目に開発した機能群の基準生産性比率の平均値と、2 回目に開発した機能群の基準生産性比率を、点線で示している。

1 回目の開発作業より、2 目の開発作業の方が、基準生産性（単位規模当たりの実績工数）に対する比率が少なくなっており、生産性が向上していることがわかる。このような類似の作業を繰り返すにつれて生産性が向上するという傾向について、他のプロジェクトでも同様の傾向があるか否かを確認するため、過去のプロジェクトデータを分析した。筆者が所属していた開発担当部門における過去の類似の開発手法を適用した 5 プロジェクトの開発工数の実績データを分析したところ、生産性と開発作業の回数の関係として、表 4-4 のような関係があることがわかった。

表 4-4　作業回数と生産性の関係

作業回数		1回目	2回目	3回目	4回目以降
基準生産性比率（倍） (対基準生産性)	上限値	1.6	1.2	1.0	1.0
	下限値	1.3	1.1	1.0	1.0

表 4-4 は、企業向けシステム開発プロジェクトにおける内部設計工程、プログラム開発及びよびプログラム単体テストまでの作業におけるプロジェクトの基準生産性を 1.0 とした場合の生産性の比率（倍）を、上限値と下限値の範囲で示している。ここで、基準生産性は、各プロジェクトにおける、3 回目以降の開発作業の生産性（単位規模当たりの実績工数）の平均値を採用した。これは、3 回目の作業以降は、繰り返し回数が増えても生産性は向上せず、一定となることが判明したためである。基準生産性比率が高いほど単位規模当たりの実績工数が大きく、生産性が悪いことを意味する。表 4-4 のデータは、同一のアプリケーション方式を採用した 5 プロジェクト（アプリケーション開発工数が 50 人月～300 人月）の生産性実績データより得られたものである。なお、プロジェクトの要員は、継続的に当該組織及び協力会社に定常的に在籍している要員であるため、組織内の基本的な開発プロセスには習熟している。

表 4-4 の集計表の対象とした 5 つのプロジェクトは、業種、業務、顧客特性、規模、要員構成等のプロジェクトの特性が異なっており、生産性実績データのばらつきが大きいため、平均値の 2 倍を越える生産性や平均値の 2 分の 1 を下回る生産性の実績値は、特定の要員のスキル等に依存する特異値として除外するなどのデータの事前処理を行った。プロジェクト数が 5 つであり、開発工数の範囲が 50 人月から 300 人月と、プロジェクトごとの相違が大きいことから、5 プロジェクトの単純平均値ではなく、プロジェクトごとの平均値の上限値と下限値で示している。このデータ分析の目的は、作業回数が増加するにつれて生産性が向上するという傾向の有無の確認であり、この視点では、5 つの全てのプロジェクトにおいて、作業回数が増えると生産性が向上する（単位規模当たりの工数実績が減少する）という傾向であった。具体的には、1 回目の作業時間が最も多く、2 回目、3 回目と作業を繰り返すことにより、単位規模当たりの作業時間が少なくなり、3 回目以降の作業の単位規模当たりの時間は、変わらなかった。あるプロジェクトを例にとると、1 回目の単位規模当たりの時間は標準の生産性に基づく工数の 1.3 倍、2 回目の時間は 1.1 倍であった。ただし、各プロジェクトの標準生産性比率はプロジェクトによって異なっており、1 回目の作業では、1.3 から 1.6 の範囲であり、2 回目の作業では、1.1 から 1.2 倍の範囲であった。

詳細 WBS 別の実績工数の分析により、作業回数と生産性の間にこのような関係がある原因は、作業の習熟による影響が大きいことが判明した。例えば、作業手順や規約の理解等の作業時間が、2 回目以降は 1 回目の作業時間の 50%以下に減少していた。なお、類似作業を 4 回以上反復しても、それ以上の生産性向上の効果は確認できなかった。また、標準化

の度合いが高いプロジェクトほど1回目の生産性が良い傾向がある。

以上の分析で判明した作業の反復による効果を「習熟度係数」として、見積式に組み込んでおくことで、反復回数を考慮しない場合に比べて、工数を精度良く見積もることができる。見積手順は以下の通りとなる。

(1) 画面単位の規模分布と開発期間より、プロジェクトにおける一人当たりの要員が担当する機能の平均工数を求める。内部設計工程や製造工程においては、各要員別に担当する機能を割り当てることが多いが、この際に、割り当てられた機能数によって、反復回数が異なる。

(2) 表 4-4 に基づき反復回数に対応した習熟度係数を生産性に乗じて補正し、以下の式により要員別の工数を算出して合計する。

工数（H） ＝ Σ［単位工数（H／KFS） × 習熟度係数 × 規模（KFS）］

（凡例）

・単位： H は時間、FS は FS (Function Scale) 法による規模の単位、KFS は 1,000 FS

・工数（H） ： プロジェクトの所要工数の合計値

・単位工数（H／KFS）：単位規模当たりの標準的な所用工数であり、標準生産性に相当する

・習熟度係数 ：作業の習熟度に対応した係数であり、標準生産性に対する倍数

・Σ： 対象システムの全ての機能の所要工数の合計を意味する

上式を適用する場合には、プロジェクトごとに、単位工数と習熟度係数を、過去の実績データを参考にして設定する。習熟度についての過去のデータの入手が困難な場合は、例えば、1回目の作業の習熟度係数を 1.5、2回目の作業の習熟度係数を 1.2、3回目以降の作業の習熟度係数を 1.0 というように暫定的な初期値を設定した上で、パイロット開発や先行して開発する機能の実績データを元に、生産性の見直しと合わせて補正を加えることによってプロジェクトとしての習熟度係数を決定する。表 4-4 の習熟度に関する数値は、同じ組織に所属する定常的要員が参画し、企業向けのシステム開発を行った場合の数値であるため、組織別に実績データを取得して見直す必要がある。

本研究で提案する変動マネジメント手法においては、生産性や習熟度、スキル係数などの各種パラメータは、初期見積りの際には暫定値を設定し、パイロット開発や先行開発作業などの実績によって数値を見直すことで、計画を修正し、計画及び変動把握の精度を向上させることでマネジメントの精度を高めるという考え方に基づくものである。本研究の変動マネジメント手法に含まれるデイリーEVM 手法では、生産性（単位工数）や習熟度を元に、詳細 WBS 別の工数計画の一括計算や、進捗率の自動計上が可能な仕組みとしており、

これにより、実績に合わせて各係数を見直した際に、容易に計画を修正可能としている。これにより、初期見積りに誤差が含まれている場合でも、早い段階で実績値を活用して補正を行うことが可能となり、計画の精度を高めることができる点が、本手法の特長であり新規性の一つである。

4.5.3.3. 規模と開発期間の考慮

開発単位（モジュールや画面）の規模が当初の想定規模より増加した場合は、反復回数が少なくなり生産性の向上が期待できなくなる。前工程の遅延等によって後工程の期間が短くなる場合も、反復回数が少なくなる。このような観点は、従来の見積り手法では考慮されていなかった。規模分布図等を利用して開発期間との関係から決まる反復回数を考慮することによって、工数の見積精度を向上させることができる。

4.5.3.4. 作業分割による工数増加の見積り式

規模の増加や前工程の遅延等により後工程の期間が圧縮された場合、スケジュールを守るために一人の担当作業を複数名で分割して期間短縮を図ることがある。この場合、仕様理解の時間や作業の摺り合わせのための相互確認等のための工数等が、作業を分割しない場合より増大する。新規加入要員に対する教育のほか、レビューワやサポート要員の追加配置も必要となる。これらのオーバーヘッドや要員数は、従来はあまり考慮されないか、プロジェクトマネジャーの経験や勘によって見積もられていた程度である。これらの影響をあらかじめ定式化しておくことで、作業分割が必要になった際の追加工数や増加要員数の見積りを的確に行え、必要な要員の投入漏れを防止できる。

そこで、作業の分割と並行実施が必要になった場合に、4.5.3.2 項で見積もった工数に対して、以下の(1)～(3)に例示する見積式によって算出する工数を加算する。なお、(1)～(3)の（例）として示す時間や人数などの各種の定数は、一例であり、教育にかける期間などは、プロジェクトごとに見直して設定するものである。

(1) 教育工数の加算

新規要員が加入した場合に、一人当たりの要員の教育に必要な工数を見込み、下記の式で所要工数を算出し、全体の所要工数に加算する。

［見積り式］

加算工数（H） ＝ 8（H／日） × 教育日数（日／人） × 追加要員数（人）

（例）

開発方式やプロジェクトのルール等に関する教育期間が 3 日間必要とした場合、

加算工数（H） ＝ 8（H／日） × 3（日／人） × 追加要員数（人）

を、教育工数として加算する。

(2) サポート工数の加算

工程の途中からから追加要員を投入する場合は業務仕様やアプリケーション方式等に関して、先にプロジェクトに参加したメンバーによる個人別のサポートが必要となるため、サポートに要する工数を加算する。

［見積り式］

加算工数（H） ＝ 8（H／日） × サポート期間（日） × 追加要員数（人）／サポート可能要員数（人）

（例）

過去のプロジェクト実績より、5 名に対して 1 名の経験者が 20 日間サポートする必要があったとすると、下記の工数を所用工数に加算する。

加算工数（H） ＝ 8（H／日） × 20（日）× 追加要員数（人） ／ 5（人）

(3) 仕様理解や整合確認の工数算出と加算

作業分割を行う場合は、分割による仕様理解などのオーバーヘッドに対応した単位規模当たりの工数を加算する。

［見積り式］

加算工数（H） ＝ 単位規模当たりのオーバーヘッド時間（H／単位規模） ×規模

（例）

仕様理解や作業引き継ぎに、1K FS 当たり 10H を要する場合、下記の式で求める。

加算工数（H） ＝ 20 （H／KFS) × 規模（KFS)

以上の(1)から(3)のような、要員追加や作業分割などの計画変更に伴う工数の加算については、経験豊富なプロジェクトマナジャーであれば、経験に基づいて工数見積りを補正することも可能であるが、経験の少ないプロジェクトマネジャーは、考慮から漏れてしまい、工数見積りが過小となることによって、スケジュールの遅延をなかなか修復できないという事態に陥ることがある。このような経験者の暗黙知となっている知識を、見積り式の形に形式知化し、プロジェクト開始前に準備しておくことによって、過去の知識を活用した適切な再見積りが可能となる。

4.5.3.5. ファストトラッキングの増加コストを考慮した要員見積り

スケジュールの遅延への対処方法として、前工程の作業が終了した機能から後工程の作業を始めるファストトラッキングが採用されることがある。ただしこの方法の採用は管理が複雑になり、様々な悪影響を及ぼす可能性が高くなる。たとえば開発とテストを平行して行う場合、テストで発生した障害修正の影響が、開発中のモジュールに影響が及び、当

該モジュールの開発が遅延すると、後続のテストに影響が及ぶといった、開発とテストが相互に影響を及ぼすことによる作業の手戻りが発生するリスクがある。また、開発中とテスト中のモジュールを別々のバージョンとして二重に管理することが必要なるというように、開発資産の構成管理が複雑になることによるデグレード等を招くこともある。これらの問題の発生を抑止するためには、障害修正に伴うリグレッションテストを行う要員を別にアサインすることや、資産管理要員を増強するなどの対策が必要となる。このような要員の追加配置等の配慮が漏れると、ファストトラッキングの採用は、スケジュール遅延や品質悪化を招く危険性がある。そこで下記の観点(1)及び(2)に基づいた追加要員数の見積り式によって、追加要員数を算出する。

ここで、下記(1)及び(2)の見積りの（例）として示している人数や規模などの各定数は、一例である。単位規模当たりの人数は、過去の実績や経験者の経験を参考にして、プロジェクトの生産性や要員の経験等を考慮して見直す必要がある。

(1) リグレッションテスト等の追加要員数の追加

［見積り式］

追加要員数（人） ＝ 1（人）／一人でテスト実施可能な規模 × ソフトウェア規模

（例）

過去の実績より、ファストトラッキングの際のリグレッションテスト要員が、15KFS の規模に対して 1 名が必要であった場合は、下記の式によって、追加要員数を算出する。

追加要員数（人） ＝ 1（人）／15KFS × ソフトウェア規模（FS）

(2) 資産管理・出荷管理の要員数算出

［見積り式］

追加要員数（人） ＝ 1（人）／一人で資産管理可能な規模 × ソフトウェア規模

（例）

過去の実績より、ファストトラッキングの際の資産管理要員として、30KFS の規模に対して 1 名の追加要員が必要であった場合は、下記の式によって、追加要員数を算出する。

追加要員数（人） ＝ 1（人）／30KFS × ソフトウェア規模（FS）×

これらの見積り式を利用することの意義は、計画の変更や要員の追加などの際に、工数加算のための観点を見積り方式に組み込んでおくことによって、考慮すべき観点が漏れることによる過小見積り等を防止することである。また、同じ見積り式を組織内のプロジェクトで継続して利用することによって、実績データに基づく統計値から見積り式の中の係数の精度を高めることが期待される。

4.5.3.6. 変動を考慮した見積方式の全体像

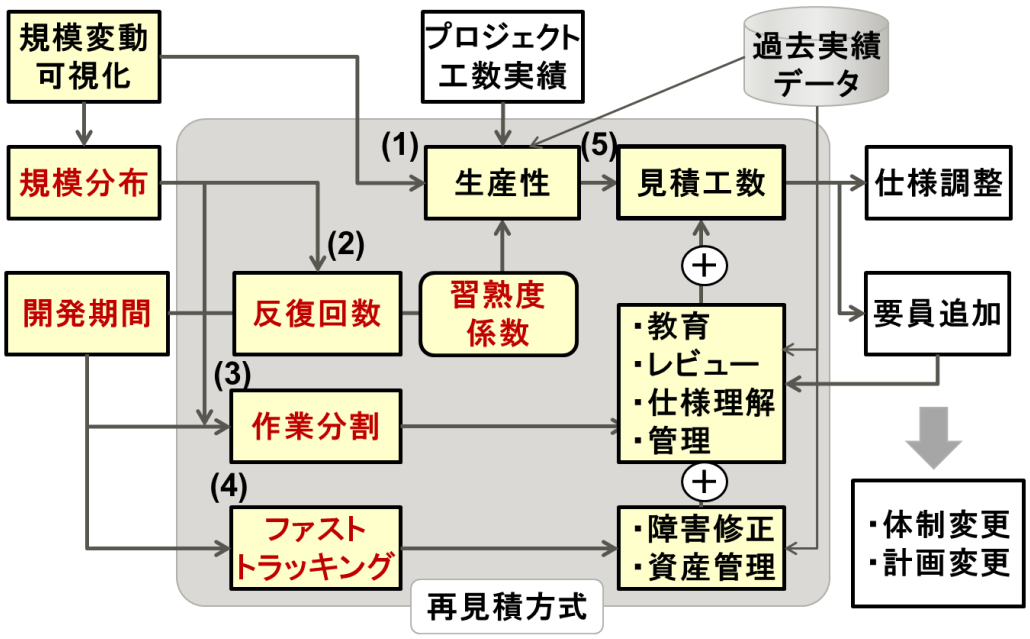


図 4-18 変動要因を考慮した工数の再見積方式（再掲）

4.5.3.1.から 4.5.3.5.に示した見積り方法を、体系的に再構成するため、**図 4-16** を図 4-18 に再掲する。図 4-18 の番号は以下の説明の番号に対応している。

規模、進捗、工数などの変動量を元に、以下の手順で工数の再見積を行う。

(1) 規模、開発期間、実績工数等からの生産性の見積り

規模、開発期間、実績工数等をインプットとして最新の規模と実績工数データを元に生産性を見直す。前節で提案した進捗・コストの変動可視化と、規模変動可視化手法を適用することにより、工数実績と最新の機能規模より、機能別、要員別の生産性の実績の把握が可能となる。なお、製造工程の生産性については、代表的な機能の設計と製造を前倒しで行うパイロット開発を計画に組み込んでおき、その実績工数を元に見直すのが望ましい。

(2) 規模分布等からの一人当たりの作業の反復回数の算出

規模分布状況と開発期間を元に反復回数を算出し、習熟度係数を用いて生産性を補正する。

(3) 作業分割を行う場合の工数加算

開発期間と規模分布を照らし合わせ、作業分割が必要と判断された場合は、作業分割に伴うオーバーヘッド分（教育、レビュー工数、仕様理解・影響確認等）を追加する。

(4) ファストトラッキングを行う場合の工数への影響の考慮

後工程との関係でファストトラッキングが必要と判断された場合は、リグレッションテスト要員や資産管理要員の追加を見込む。

(5) 合計工数の算出

機能別のソフトウェア規模に、(1)で求めた生産性と、(2)から求めた習熟度を乗じ、(3)と(4)で考慮した工数を加算することによって、工数を算出する。

4.6. 変動に対するマネジメント手法のまとめ

ここで、本章で提案した変動の可視化と再見積り方式を含むコントロール手法を合わせた変動マネジメント手法の全体像を図 4-19 に示す。

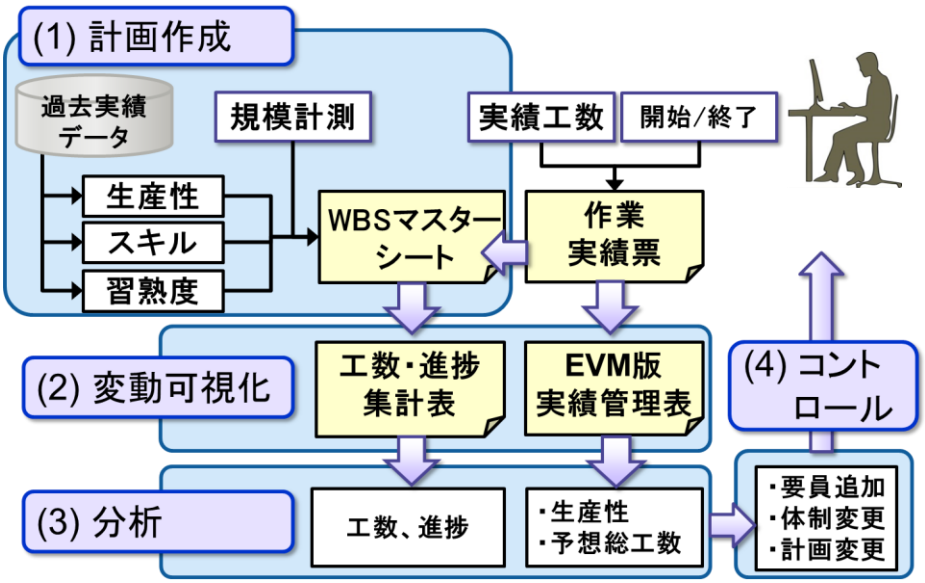


図 4-19　変動マネジメント手法の全体像

次に、図 4-19 の変動マネジメント手法に基づく具体的なコントロールの実施パターンを、図 4-20 に示す。規模、進捗・コストの変動の可視化手法と変動に対する見積方式を活用し、原因の種類に対応したプロジェクトのコントロールの方法をパターン化することにより、変動のマネジメントに関する知識を形式知化したものである。

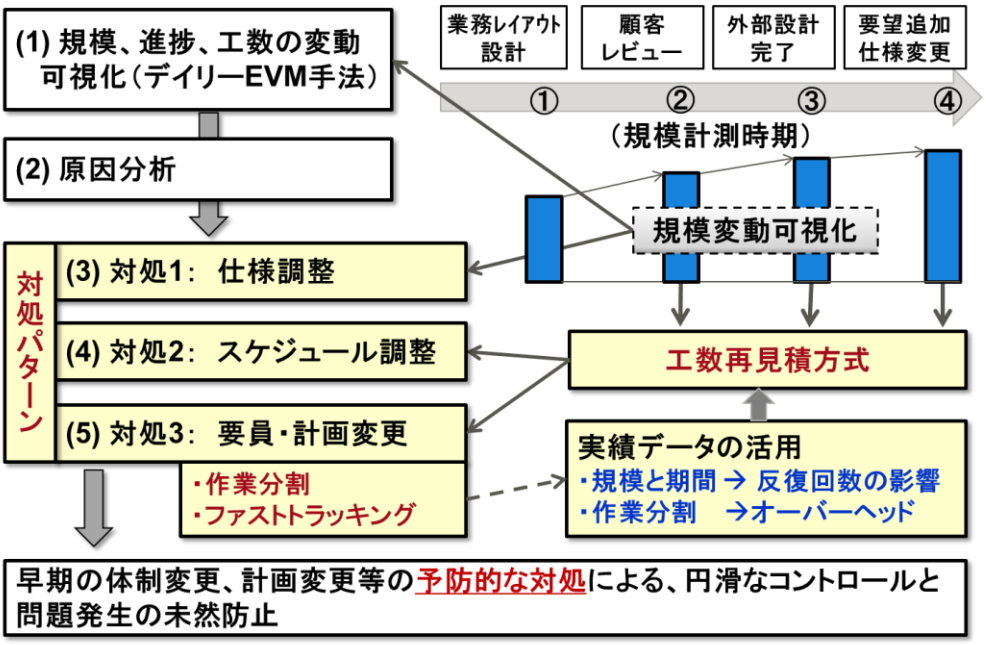


図 4-20　変動可視化によるコントロールのパターン

以下の手順で見積結果を活用したマネジメントを行う。各手順の番号は図 4-20 の番号に対応している。

(1) 規模、進捗、個数の変動可視化

毎日の実績データから、ソフトウェア規模、進捗、コストの変動状況を可視化する。この際、プロジェクトオーナーを含むステークホルダとの間で、規模変動に応じて仕様調整や費用、スケジュールの見直しを行うことの事前合意を得ておくことが重要である。

(2) 原因分析

設計途中での規模、進捗、コスト（工数）の変動状況を可視化し、変動の大きい機能や担当者、WBS 対する重点的な監視を行う。変動可視化用の各種のドキュメントや指標を活用することで、進捗遅延などの原因分析を行い、計画との差異を解消するめに必要な対策を行う。規模の変動や期間の変更が発生した場合は、デイリーEVM 主要によるコスト予測に加えて、図 4-18 の再見積方式等を活用し、要員追加やスケジュール見直し等の対策を実施する。この際の、変動の原因に応じた対策を、(3)～(5)の対処1～対策 3 などの対処方法としてパターン化することにより、プロジェクトマネジャーに依存しがちなコントールに関する暗黙知を形式知化することが可能となる。

(3) 仕様調整

規模の増加が判明した場合は、ステークホルダとの間での仕様調整を早めに行うことが重要である。仕様の追加や変更が必要な場合は、そのためのコストを見積もり、予算内に収めるために、コストと機能のトレードオフによる判断を行う必要がある。この時に、コストの再見積り手法が有効に活用できる。

(4) スケジュール調整

仕様調整を行っても規模増加が避けられない場合や、生産性が計画値よりも低下した場合などには、作業分割等を考慮した見積りや要員の手配などの予防的対策を早めに行うことで、後工程の混乱を最小限するためのコントロールを行う。ステークホルダとの間で予算の追加やスケジュール変更等の調整を行った上で、図 4-18 の再見積方式にしたがって増加工数を算出する。スケジュールやリリース時期の調整が可能な場合には、プロジェクト計画の変更が必要となる。

(5) 予算、要員、計画の修正

内部設計からプログラム開発及び単体テストまでの工程においては、規模、要員別スキル、反復回数等を考慮して画面などの開発単位ごとに最適な要員を割り当てる。

スケジュールの延伸が不可能な場合は、作業分割やファストトラッキングの必要性を判断し、本見積方式によりさらなる追加工数を算出し、要員追加、体制変更およびプロジェクト計画の見直しを行う。

上記の変動マネジメント手法のポイントは、機能や人別の詳細な WBS 単位の変動を日次で可視化することと、プロジェクトの変動要因に応じた代表的な対応策を過去の経験を元にパターン化し、対策に必要となる実績データの分析方法や見積方式等のコントールに必要な知識を形式知化し、準備しておくことである。

従来は、計画との差異に対するコントロールについては、進捗遅延などの問題が発生してから、その都度、プロジェクトマネジャーの経験による見積りや属人的な判断を加味して行われていた。見積りのためのデータの蓄積も不十分であった。このように、コントロールに関するノウハウは、個々のプロジェクトマネジャーの経験に基づく暗黙知となっていることが多く、知識の活用や継承が難しかった。

本手法を適用することにより、実績データに基づいて、規模の変動や機能別、人別、WSBS 別の変動を日次で可視化し、生産性や予測工数等を継続的に把握することができる。これにより、変動の早期発見と、その要因となる機能、人、作業の特定が容易となる。さらに、原因に応じた対処のパターンや工数の再見積方式を利用することによって、顧客交渉や要員追加等の対策を早い段階で実施することで、スケジュール遅延等の問題を未然に防止するための予防的なプロジェクトマネジメントが可能となる。

以上の変動マネジメント手法によって、過去の知識を活用したプロジェクトの円滑なコントロールが可能となることを、次章で具体的なプロジェクト適用事例によって示す。

5. プロジェクトへの適用による評価

本章では、第 4 章で提案した変動マネジメント手法である「デイリーEVM 手法」による変動の可視化、規模変動の可視化、及び変動のコントロールについて、ソフトウェア開発プロジェクトに適用した事例を用い、デザインサイエンスの方法論 (Hevner et al 2004) により、その有効性を示す。始めに、適用事例による評価方法について提示する。

5.1. 評価方法

以下に、解決すべき課題、前提条件、評価手法、評価観点について示す。

(1) 解決すべきプロジェクトマネジメントの課題

進捗やコストの実績を、週や月の管理サイクルで、サブシステムやチームを単位とした粒度で把握するといった従来の俯瞰的な監視方法では、機能別や担当者別の作業遅延等の問題の検出、原因究明や対策が遅れ、問題が拡大してしまうリスクがある。的確に変動を把握可能とするためには、収集するデータの粒度をなるべく詳細な粒度とした上で、問題箇所の検出を支援する仕組みを有するのが望ましい。ただし、データの粒度を収集・分析などの運用負荷が高くなるため実行は困難である。また、変動の原因分析や対処方法などの、プロジェクトマネジャーの知識の多くは暗黙知となっており、継承することが難しい。

(2) プロジェクトの前提条件

変動マネジメント手法の適用対象プロジェクトは、企業内の情報システム開発等の、要件定義に基づいて策定した計画に沿って推進するプロジェクトを前提としている。また、対象となる組織は、企業の情報システムの開発を繰り返し実施し、過去の実績データの活用が可能な組織を想定している。以下に例示するデイリーEVM 手法等を適用したプロジェクトは、これらの前提条件を満たすプロジェクトである。

(3) 検証方法

デザインサイエンスの方法論に基づき、提案した手法でのねらいや仕組みについて既存の方法と比較し、提案手法によって実現可能となった点を適用結果の評価で示すことにより、提案した手法の有効性を検証する。

(4) 評価の観点

提案した変動マネジメント手法の有効性を、以下の観点で評価する。

(a) 以下の各観点でのプロジェクトマネジメントの知識の形式知化によるマネジメントの

円滑化と知識継承の効果

1) WBS 作成方法の知識の形式知化による効果

2) 工数計画作成の知識の形式知化と自動化による効果

3) 進捗・コストの変動把握方法の自動化による効果

4) 規模の変動の定期的、連続的把握による効果

5) 変動の原因に関係する工程や作業を特定する上での実績データの活用効果

6) 変動のコントロール方法の形式知化による知識活用の効果

7) 計画変更のための工数の再見積りによる効果

8) データの蓄積・活用に関する効果

(b) 変動把握のマネジメント精度向上の効果

1) 変動把握の管理サイクル

2) WBS 粒度及びばらつき

3) データ収集の種類

(c) 変動把握のデータ収集・分析に関する運用負荷の低減の効果

1) 計画作成負荷の低減

2) 実績集計時間の抑制

3) 実績把握の情報源による運用負荷経験の効果

5.2. プロジェクト概要

以下に、変動マネジメント手法適用を適用したプロジェクトの概要を示す。

・業務機能　　：　金融関係企業の契約審査業務

・システム形態：　Web システム

・開発プロセスモデル：　ウォーターフォール型開発プロセスモデル

・開発期間：外部設計～結合テストまでで 9 ヶ月

・規模：　　当初見積規模　21 KFS、最終規模（実績）　29.4 KFS

（本プロジェクトではソフトウェア規模を FS (Function Scale) 法で計測）

・要員数：　当初 10 名、ピーク時 29 名

・体制：　図 5-1 に示す。

プロジェクトマネジャーは、過去のプロジェクトにおいては、開発チームリーダとしての 3 年間のプロジェクトマネジメント経験はあるが、外部設計の工程からのプロジェクト全体工程のマネジメントを担当するのは始めてであった。

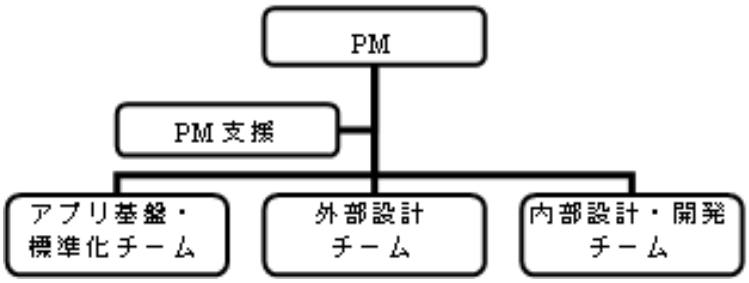


図 5-1　プロジェクト体制図（概要）

・スケジュール（プロジェクト開始時）：

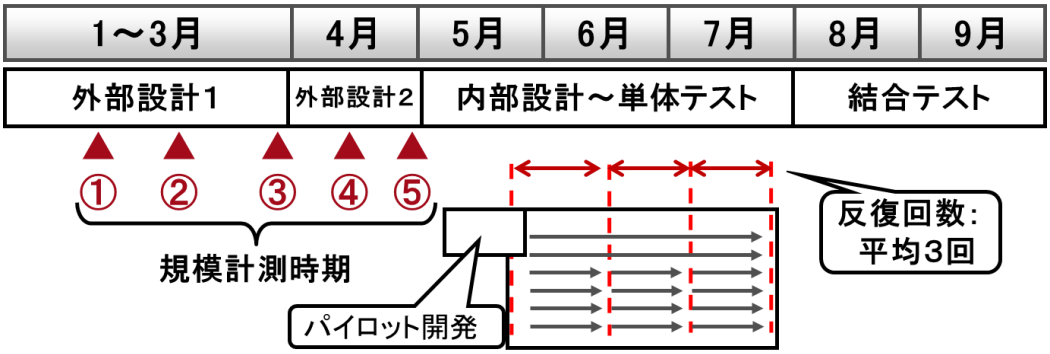


図 5-2　当初スケジュール（外部設計～結合テスト）

図 5-2 に、外部設計から結合テスト工程までの当初のスケジュールを示す。月数は相対月数である。当初の計画では、内部設計から単体テストまでの開発工程は 3 カ月で終了する計画を策定していた。反復回数は平均 3 回を想定して生産性を見積もった。また、①から⑤の数字は定期的な規模計測と分析を行うタイミングを示したものである。

5.3. 変動マネジメント手法の活用結果

5.3.1. 計画策定への活用

最初に、変動マネジメント手法の中の計画策定の部分について示す。プロジェクトの WBS を作成する際には、詳細 WBS が記入された「WBS マスターシート」の標準テンプレートを参照し、一部の順番を変えるなどの検討を加えて本プロジェクトの WBS を設計した。標準の WBS を再利用することにより、プロジェクトごとに個別に WBS を策定する方法と比較し、経験の少ないプロジェクトマネジャーでも漏れや無駄のない WBS を作成することができた。例えば、「仕様理解」や「Q&A（質問）」、「テストデータ作成」、「レビューによる

指摘後のドキュメント修正」などの WBS は、従来の数日単位の粒度の WBS では挙げられなかった WBS であるが、本手法のテンプレートを利用することで、漏れなく作成できた。また、各種の設計ドキュメントなども、経験が少ないプロジェクトマネジャーでは、漏れなく洗い出すことは難しかったが、重複や漏れなく抽出することができた。

次に、開発方式が類似している過去のプロジェクトの実績データに基づく生産性等を参考にして単位工数を見直し、プロジェクト用の WBS マスターシートのテンプレートを完成させた。本プロジェクトでは、過去の開発方式が類似したプロジェクトで適用した WBS マスターシートを流用して見直すことにより、既存手法と同じ程度の 1.5 日程度の期間で、WBS と単位工数を設定することができた。

作成したプロジェクト用の WBS マスターシートに対して、機能別の規模や生産性、担当者のスキル係数等を一覧表化した「機能・担当者別パラメータ一覧」に基づいて、各パラメータの値を設定することにより、詳細 WBS 毎の基準工数が自動計算され、全ての機能の工数計画を一括で作成することができた。

WBS 別の基準工数は、4.3 節で示した下記の式で計算される。

基準工数 (H)

＝ 単位工数（H／単位規模）× 規模 × スキル係数 × 習熟度係数 × 工程係数・・①

これに対して、本手法によらない手作業による方法では、機能別、WBS 別の工数計画を作成するためには以下の(a)～(d)の手順で設定する必要があった。

(a) 機能別の開発工数の見積り

プロジェクトの平均的生産性を元に、以下の式により機能別の開発工数を見積もる。

機能別開発工数 (H) ＝ 規模 ÷ 生産性（規模／H） ・・・②

この場合は、式①におけるスキルや習熟度等は考慮されないため、この方法で見積もった工数計画では、人による生産性の違いなどにより、見積りと実績工数の乖離が大きくなる場合がある。

(b) 上記で見積もった機能別の開発工数を、過去の統計値等から求めた工程別の工数比率で、各工程に按分する。

(c) 各工程内の WBS 別の工数については、プロジェクトマネジャーやチームリーダが、過去の経験を元に WBS 別に見積り工数を割り当てる。

従来の一般的な企業内システム等の開発では、一つの工程の WBS の数は、設計、作成、自己レビュー、第三者レビュー、工程完了判定等の 5 個程度であることが多い。したがって、内部設計、プログラミング、プログラム単体テストの 3 工程分の WBS 数は、5 個×3 工程＝15 より、15 個程度になる。

(d) 上記の(a)から(c)の作業を、機能の数だけ繰り返す。

1 機能の内部設計からプログラム単体テストまでの 3 工程分の工数計画を 15 分で作成可

能と想定すると、25 機能分の工数計画作成には、6 時間程度を要する。

これに対して、デイリーEVM 手法の場合は、内部設計からプログラム単体テストの 3 工程における WBS 数の合計は、約 130 個となり、上記で想定した 15 個の 8 倍以上に増加する。このため、従来と同じ手作業による方法で工数計画を作成するためには、単純計算では上記の 8 倍の 48 時間以上を必要とするという計算になる。1 日の作業時間を 8H とすると 6 日間を要することになる。また、規模や生産性の変動が発生した場合に、一度作成した工数計画を修正するためには、上記と同様の手順を繰り返す必要があり、工数計画の修正時にも、初期計画の作成と同程度の時間を要するため、頻繁に工数計画を見直すのは難しかった。

本手法（デイリーEVM 手法）を適用した場合、WBS マスターシートのプロジェクト用テンプレートの作成に 3 時間を要したが、機能別、担当者別の生産性や習熟度の一覧を 2 時間で作成することによって、5 時間で全ての機能別の工数計画を一括で生成することができた。また、規模の変動や生産性の見直し等による計画修正を行う場合も、一覧表の規模、生産性、習熟度、スキル係数等を修正するだけで、機能別、担当者の工数計画を修正することが可能となった。

以上のように、担当者別の生産生等を考慮した精度の高い計画を、従来の手作業による工数計画の策定方法より負荷を増やすことなく作成することができた。

既存手法では、人別や反復回数を考慮しないため、プロジェクトの標準生産性に比べて生産性の低い担当者については、計画と実績の工数の乖離が大きくなる場合がある。本事例のプロジェクトでは、スキル係数が 1.7 などのメンバーが数名含まれていたため、一律の生産性を適用した場合、70%の計画と実績の工数の乖離が発生した可能性がある。本プロジェクトでは、図 5-2 のスケジュールに示すように、「パイロット開発工程」を設け、代表的な機能の開発を先に行うことによって実績データを取得し、単位工数や生産性の標準値を補正することで、工数計画の精度を高めることができた。具体的には、初期の工数計画では最大で担当者により 30%の計画と実績の差異が発生したが、パイロット開発の実績データや個人別の実績生産性等を元に計画を見直した後には、人別の最大で 10%、平均で 5%の差異に止まった。従来の手法では、計画の修正を行う際には、初期計画と同じ程度の負荷が必要であったため、計画と実績の乖離が大きくなっても計画を見直すことが難しかった。これに対して本手法では、一つの WBS マスターシートのテンプレートを修正するだけで、全ての機能及び担当者別の計画を一括で修正できるため、必要な時に、容易に計画を修正することが可能となった。本手法を適用した他のプロジェクトにおいても、パイロット開発や先行機能の開発実績データを元に「単位工数」を見直し、工数計画を適時に補正することが可能となった。これによって、工数計画の精度を向上させることができた。

以上のように WBS 作成や工数計画作成のための過去の知識を、形式知として活用することが可能となった。

5.3.2. 変動可視化情報の活用と分析

プロジェクトの実行段階では、担当者が作業実績票に入力した実績工数や開始・終了状態が WBS マスターシートに毎日累積され、進捗率が自動的に計上される。これらの情報を元に出力される各種の変動可視化用のシートによって毎日の変動をリアルタイムに確認することが可能となり、異常の早期発見や対処が可能となる。この際の担当者一人当たりのデータ入力に必要な時間は、毎日数分以内であり、既存のマネジメント方法と変わらない負荷で日々のデータ収集が可能であることを確認できた。

以下に変動可視化情報の活用事例を示す。

基本情報			内部設計工程						
システム情報			担当者	作業時間			生産性（内部設計）	進捗率	ドキュメント進捗率
機能ID	機能名	規模		基準作業時間	実績作業時間	基準作業時間消化率			
合計		28633		2361.4	3274.3				
平均		1145.3		94.5	131.0	139%	0.0	90%	92%
ABCAA03	機能A	1512	A	140.2	268.9	192%	17.8	100%	100%
ABCAA04	機能B	2723	B	230.7	339.0	147%	12.4	100%	100%
ABCAA05	機能C	1055	C	102.1	121.2	119%	11.5	96%	85%
ABCAA06	機能D	461	D	47.0	0.3	1%	0.1	20%	0%
ABCAB03	機能E	1235	E	124.1	155.3	125%	12.6	90%	100%
ABCAB05	機能F	1677	F	129.6	214.1	165%	12.8	86%	88%
ABCAB07	機能G	1250	G	125.6	180.9	144%	14.5	79%	80%
ABCAB08	機能H	1783	H	151.4	153.6	101%	8.6	90%	100%
ABCAB09	機能I	1839	I	230.2	162.8	71%	8.9	85%	68%
ABCAB12	機能J	1033	J	61.9	60.2	97%	5.8	100%	100%

図 5-3　工数・進捗集計表の例（再掲）

図 4-8 の「工数・進捗集計表」を図 5-3 に再掲する。機能別の基準作業時間消化率（基準工数に対する実績工数の比率）や進捗率を一覧表示している。予定と実績の差異の比率が一定の比率を越えた場合に背景色が変わるようにしておくことで、変動の大小を視覚的に捉えやすくなり、重点的なマネジメントに役立つようにしている。本プジェクトでは、基準作業時間消化率が 120%以上となった場合は数値蘭の背景に薄い網掛けを行い、150%以上となった場合は濃い網掛けを行うよう、しきい値を設定した。これによって、濃い網掛けとなった機能を中心に重点的に変動の原因確認や早期の対策を実施し、薄い網掛けとなっている機能については、状況の変化を注視するというように、効果的にマネジメントを行うことができるようになった。

この「工数・進捗集計表」の活用場面の一例として、内部設計工程において、特定の機能の基準工数に対する実績工数の比率がしきい値を超えた場合を取り上げる。図 5-3 の工

数・進捗集計表において、ある機能の基準作業時間消化率の背景色が濃い網駆けに変わったことにより、予定よりも実績工数が大幅に増加していることが判明した。該当機能の WBS マスターシートの実績値の例を図 5-4 に示す。

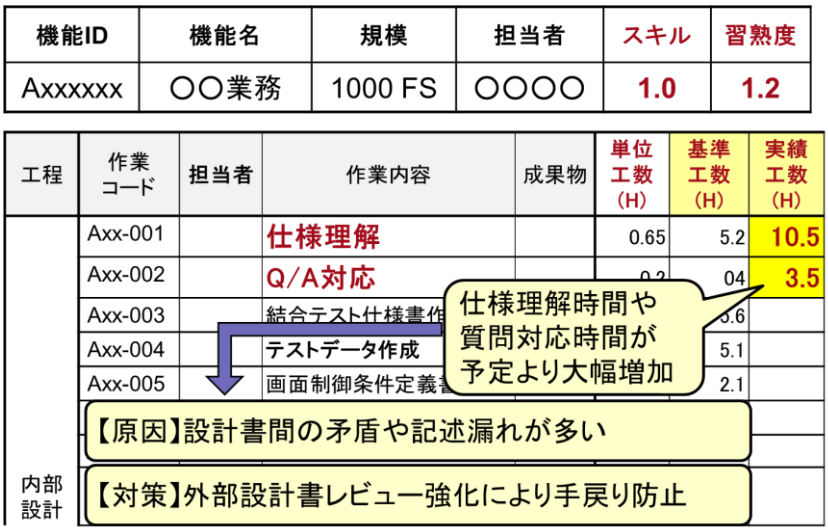


図 5-4　WBS マスターシートに集計された実績値の状況（例１）

WBS マスターシートの実績データの欄を確認すると、外部設計書に関する「仕様理解」や「Q&A（質問対応）」の時間が基準工数より増加していることなどが読み取れた。当該機能の設計書を元に担当者にヒアリングした結果、外部設計書の記述が曖昧であり、仕様に関する不明点が多いことが判明したため、有識者による外部設計書のレビューを強化することによって、外部設計書の記述品質を改善することで手戻りを未然に防止するための対策を講じた。

このように、プロジェクトマネジメントでは、レビュー強化やテストの再実施など、一度終了した WBS を再実施することが必要となる場合があるが、本手法では、そのような場合でも工数計画を容易に修正できるため、途中での WBS の追加等も行うことができる。このケースでは、WBS のコードに再実行を意味する記号を追加することで、WBS の追加と実績収集や進捗把握を行った。この場合は、レビューを実施する担当者と、レビューを受ける担当者の双方の WBS マスターシートに、新たな WBS を追加した。

次に、レビュー後の修正時間が基準工数より増加した機能の WBS マスターシートの例を図 5-5 に示す。このケースでは、記述内容の誤りが多いなど、担当者のスキル不足が原因であることが原因として判明したことにより、スキル不足を補うための内部設計の個別レビューを強化する対策を実施することとした。この場合も、レビュー実施者とレビューを受ける担当者の双方の WBS マスターシートの基準工数に追加工数を加算することで、工数計画を変更した。

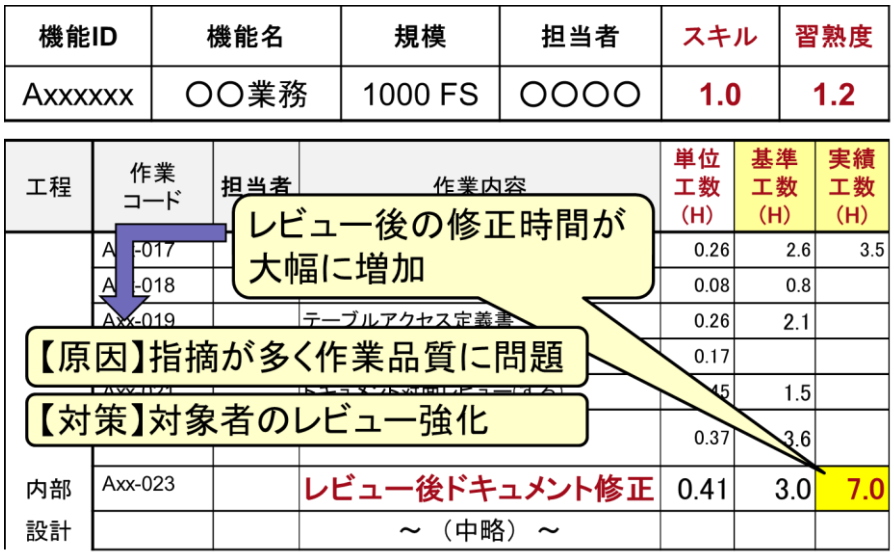


図 5-5　WBS マスターシートに集計された実績値の状況（例２）

従来の、チームやプロジェクト全体などの単位での粗い粒度の WBS に基づく進捗管理や、報告に基づく進捗把握の方法では、このような機能別や人別、作業別の異常の検知や、実績データを活用した問題の原因分析は難しかったが、本手法によって、機能や人別の変動を早期に検出し、実績データを元にして問題発生の原因となる工程や WBS の特定等を行うことができ、適切な対応方法を判断することが可能となった。

5.3.3. 規模変動の可視化

本プロジェクトでは、ソフトウェア規模を FS 法で測定し、プロジェクト計画策定の際に、規模を再測定する時期と、その情報により仕様や費用の見直し等を行うことについて関係者の了解を得た。また、設計担当者ごとに仕様変更等が発生した都度、FS 値の差分を再測定するようにすることで、早めに仕様調整などに活用できるようにした。その上で、主要な時点でプロジェクト全体の規模計測結果の集計を行い、仕様の調整や要員再配置などに活用するよう計画した。

表 5-1 に主な時点の FS 計測値の履歴を示す。No.欄の①から⑤は、図 5-2 のスケジュール中に示した規模計測時点に対応しており、各時点の FS 値と、①の初期計測値に対する FS 値の比率（%）を示している。

ここで「外部設計 1」は利用者視点での設計、「外部設計 2」はデータアクセスや外部インタフェースを含む詳細設計という 2 段階に分け、早期に顧客視点のレビューを進められるようにした。

表 5-1　規模の変動状況

No。	計測時点	FS 値 (FS)	初期計測値に対する比率
①	初期計測	21,242	―
②-1	外部設計 1（主要機能）	26,185	123.3 %
②-2	仕様調整後	21,335	100.4 %
③	外部設計 1 完了時	22,860	107.6 %
④	外部設計 2（中間時点）	26,230	123.5 %
⑤	外部設計 2 完了時	29,423	138.5 %

図 5-6 に表 5-1 の規模変動の状況を図示する。

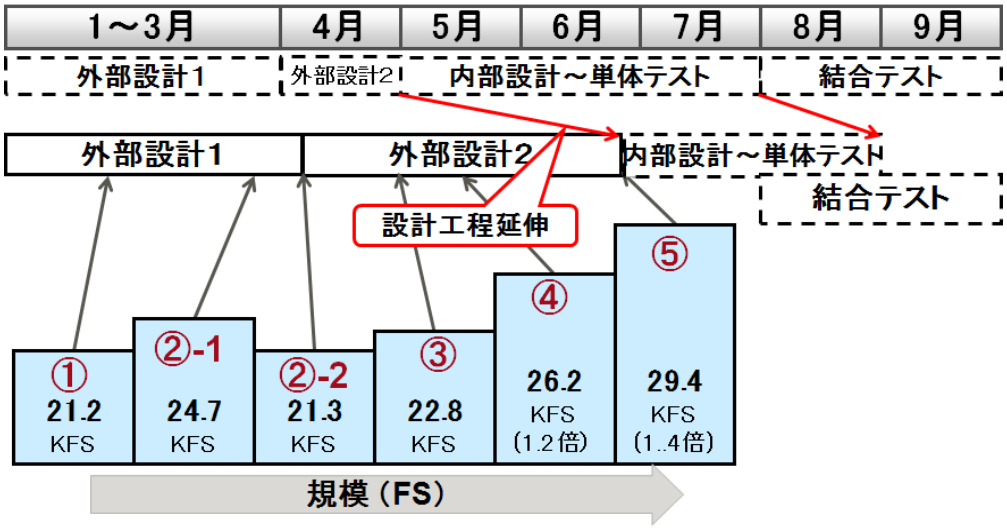


図 5-6　規模の増加状況の可視化

規模の計測時期については、以下のように計画した。

(1) 要件定義工程では画面レイアウトは未作成であったため、外部設計 1 の工程の初期に画面レイアウト作成を行う計画とし、その時点で初期の FS 法による規模計測を実施（①）

(2) 外部設計 1 の中間で、画面レイアウトと主要な処理（検索、表示、登録、更新等）が一旦確定した時点（②-1）

（②-2 は、仕様調整後の規模を表す）

(3) 外部設計 1 のレビュー完了時（③）

(4) 外部設計 2 の中間（④）と完了時（⑤）

中間時点で計測するのは、工程が完了してからでは、次工程に向けた要員追加の手配や準備が間に合わなくなるためである。

以上のように、事前に計画した時期（①～⑤）に規模計測を行い、変動状況を継続的に可視化した。これにより、増加後の規模に対する開発工数やスケジュールへの影響を早めに予測できるため、仕様の調整やスケジュールの調整を行い易くなった。

5.3.4. 規模変動への対処

ここで、表 5-1 や図 5-6 に示す規模変動の背景と対処方法について述べる。

図 5-6 の通り、外部設計 1 の中間の時点（②-1）で、規模が当初の 21.2K FS から 24.7K FS となり、123.3%に増加した。これは、顧客レビュー中に、要件定義段階では不明確であった仕様が詳細化され、機能的な要望等が発生したことによるものである。そこで、規模増加に対する費用見積り等を元に、プロジェクトオーナーとの間で仕様削減のための調整を行った結果、②-2 に示すように、ほぼ初期計測値に近い規模に抑制することができた。

これは、変動可視化手法における「EVM 版実績管理表」等によって、規模増加によるコスト増加や進捗への影響、及びプロジェクト完了時のコスト予測などを、ステークホルダとの間で定期的に共有し、合意形成を円滑に進めることができた効果によるものである。

ただし、その後は規模が増加している。この背景や対処については、以下で説明する。

5.3.5. 進捗、スケジュール及びコストの変動管理

図 5-7 に変動可視化手法における「EVM 版実績管理表」による進捗及びスケジュールの変動把握方法の一例を示す。

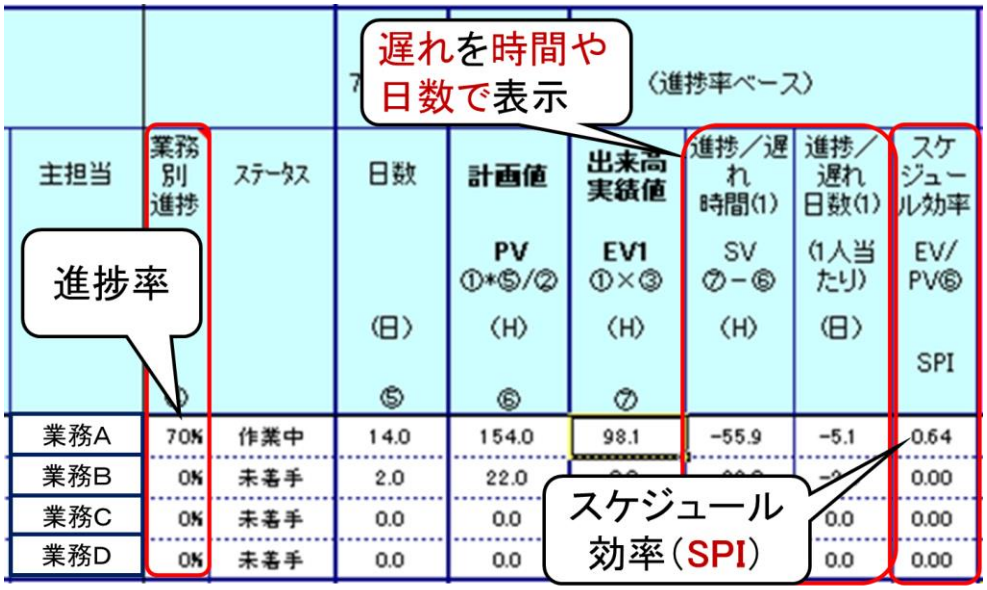


図 5-7　EVM 版実績管理表によるスケジュールの変動把握

このシートによって、日々の進捗を日数単位で把握することができる。EVM 版実績管理表によって、機能別のスケジュールの遅れが毎日、日数（小数点以下１桁）で表示される。EVM における指標値としての SPI（スケジュール効率指数）や CPI（コスト効率指数）（図では省略）なども毎日更新される。これによって、機能や担当者別の問題点の把握や対処が可能となった。

上記の EVM 版実績管理表によって、機能別のスケジュールの遅れが毎日、日数（小数点以下１桁）で表示される。EVM における指標値としての SPI や CPI（図では省略）なども毎日更新される。これによって、機能や担当者別の生産性等を確認することが可能となり、機能や人別の問題点の把握や対処を容易に行うことができるようになった。

次に、図 5-8 に、EVM 版実績管理表によるコスト予測の例を示す。

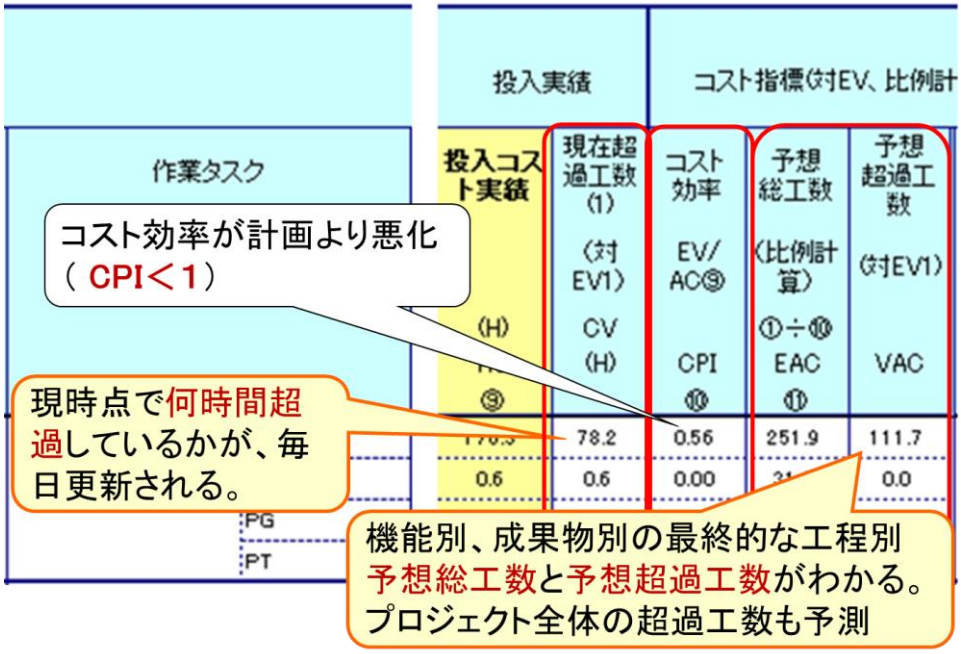


図 5-8　EVM 版実績管理表によるコスト予測

本プロジェトでは、仕様の削減について調整を行った後に新たな要件の漏れが判明し、仕様の追加に伴う規模増加により、進捗やコストの計画との乖離が大きくなり始めた。これは、外部システム連携に関する新たな要件が判明し、業務運用の根幹にかかわることが明らかになったためである。

プロジェクトオーナーによって、費用を追加してでも実現する必要があるという判断が行われた。この際に、図 5-8 に示すように、EVM 版実績管理表によって、現時点でのスケジュール超過や完了時の予想総コストが毎日表示されるなど、ステークホルダの間で状況を常に共有できていたため、仕様や費用の調整等の合意形成を効果的に進めることができた。

本手法を適用せず、規模の増加を察知することができなかった場合には、表 5-1 や図 5-6 に示すように、仕様変更後に 138%の規模増加が発生し、これによって最大で 38%のコスト超過やスケジュール遅延を招いた可能性がある。本プロジェクトでは、外部設計工程において規模増加の量を把握できたため、内部設計以降の計画を見直し、適時に要員追加等の対策を実施したことによって、このようなスケジュールやコストの超過を防止することができた。

以上のように、日々の規模や進捗、コストの変動を可視化することにより、プロジェクトマネジャーやメンバー及び各ステークホルダが早期に問題点を把握することができ、これによって問題を先送りせず早期に対策を構ずることで、プロジェクトの円滑なマネジメントを行うことが可能となった。

従来の EVM の適用方法がサブシステムやチーム別などの進捗やコストの俯瞰的な把握に止まっていたのに対して、本手法により機能別や担当者別のスケジュール効率やコスト効率、完成時総コスト等を毎日確認可能となった。これによりステークホルダとの間でのプロジェクトの将来の見込みの共有や、先を見越した対策などの予防的なマネジメントを行うことができたことがプロジェクトの成功の大きい要因であった。

5.3.6. 規模増加による影響の分析

プロジェクトオーナーから費用追加については了承されたが、システムのリリース時期は、ビジネスへの影響を鑑みて変更できない状況であったため、稼働時期は変更せずに当初の納期を守ることが条件となった。この場合、規模増加に伴って、一人当たりの反復回数が減少するため、変動を考慮した見積り手法によって、生産性がどの程度悪化するかを事前に見積もることができた。

さらに本プロジェクトでは、仕様追加等のために設計期間を 1.5 か月延長したため、後工程のプログラム開発や結合テストの期間を短縮する必要があった。この規模の増加と期間の短縮のための要員追加や作業分担の見直しが必要となった。

そこで、6 章で示した、変動を考慮した工数見積り手法を含むコントロール手法によって、規模増加や期間短縮による生産性の見直しと工数見積を行い、見直した生産性や習熟度係数を用いて、変動管理手法における計画の自動作成を活用した新たなスケジュールを作成した。この見積りやスケジュールをプロジェクトオーナーに提示することで、予算追加等の承認を得ることができた。以下に具体的な再見積りの結果や対処方法を示す。

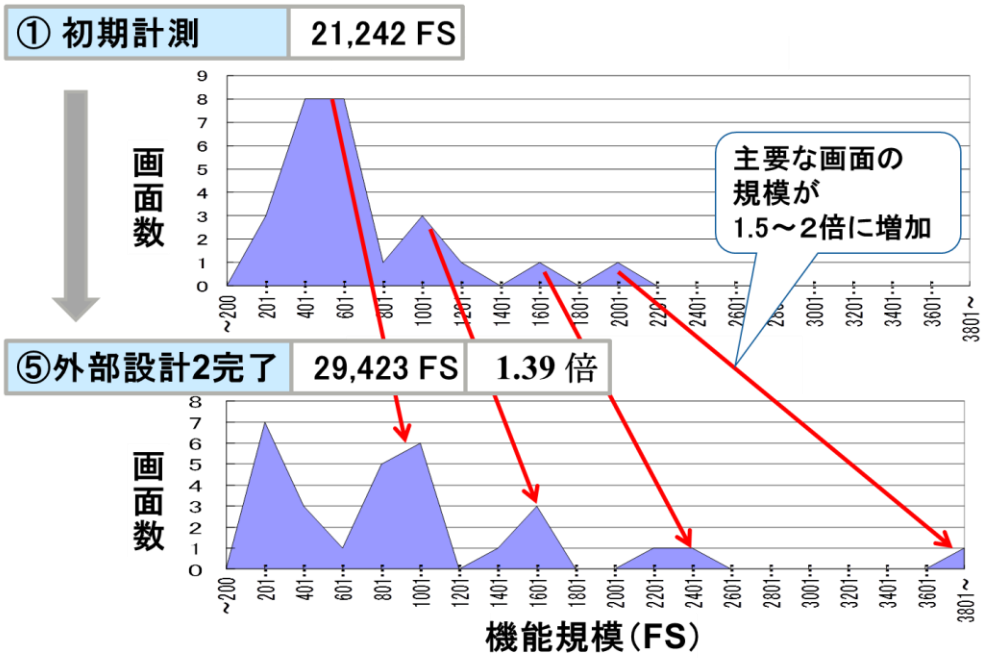


図 5-9　機能規模の変動結果

図 5-9 に、初期計測時（計測時期①）と外部設計 2 完了後（計測時期⑤）の FS 法による規模分布図を対比して示す。過去の統計データによると、平均的な Web システムでは 1 画面当たりが 400 FS～600 FS 程度のものが多い。しかし本システムは、最終的に図 5-9 のように主要な画面の規模が 1.5 倍から 2 倍に増加し、2,000 FS 以上や 3,000 FS 以上の非常に規模が大きく複雑な画面が含まれるものとなった。

初期計測時（①）は 400 FS～600 FS の規模の画面が多かったが、外部設計完了時（⑤）には、1,000 FS、2,000 FS、3,000 FS 以上の規模の画面が増加した。これらの画面の規模は、社内の実績データによる Web システムの平均規模 400 FS～600 FS と比較して 3 倍～4 倍の規模である。このため、反復回数は当初の見込の 3 回から、2 回または 1 回に減少し、生産性が画面によって 10%から 30%悪化するという見積りとなった。さらに 2,000 FS を超える画面は開発期間中に完成できない見通しとなった。

以上のような変動により計画の大幅な見直しが必要となったが、変動可視化手法によって規模分布や工数、進捗の変動状況を継続的に可視化していたため、影響を早い段階で察知でき、以降で述べる要員追加を含む計画変更の対策に、早めに着手することができた。以下に具体的な計画の見直し方法について説明する。

5.3.7. 作業分割について

結合テスト工程以降のスケジュールを考慮すると、製造期間を延伸することは不可能と判断されたため、開発単位内で作業分割を行うこととした。

具体的には、システムの画面を中心としたビュー層及びコントロール層と、データベースアクセスを行うモデル層を別の要員で分担して並行的に作業を進めることとした。この作業分割による追加工数は、前章の再見積方式によって見積もることができた。

5.3.8. ファストトラッキングによる増加コストの見積り

本プロジェクトでは、プログラム設計工程以降の期間の短縮が必要となったため、作業分割を行っても、テスト工程が圧迫され、実施期間が不足してしまうことが判明し、単体テストが終了した画面から結合テストを開始するという、ファストトラッキングを採用せざるを得なくなった。そこで、ファストトラッキングによる増加工数や追加要員数も本見積方式にしたがって見積もった。

5.3.9. 要員配置やプロジェクト計画の見直し

本見積方式を元に算出した必要工数と開発期間や規模分布より、当初計画の要員数 10 名を約 3 倍の 29 名に増員する必要があることが判明した。

規模が増大し始めた段階から逐次、本見積式に従って追加要員数を見積もっていたため、早い段階から要員の手配を進め、内部設計開始前までには、必要な要員の追加と再配置を含む体制変更及びプロジェクト計画の変更を実施することができた。

本プロジェクトでの再見積方式での考え方を以下に整理しておく。

(1) 規模増加により一人当たりの反復回数が 2 回以下に減少したことを反映し、担当者ごとに反復回数に応じて工数が標準工数よりも 1.1 倍から 1.3 倍に増加すると見積もった。
(2) 画面単位の規模増大と開発期間短縮への対策として、ビュー層、モデル層等に分割して別要員で平行開発したため、追加要員一人当たりの仕様理解時間を 20 時間見込むなど、オーバーヘッド分を工数見積りに反映した。
(3) 新規要員の加入に伴う教育コストの増加分を追加した。
(4) 6 名の追加要員につき 1 名のレビューワ追加や、チーム分割に伴うリーダー及び PMO 要員や資産管理要員の増員を見込んだ。

このような規模増加や期間短縮に伴う要員の追加配置等の対処は特別なことではないが、規模増加や進捗、生産性の変動把握やその原因分析が不十分なまま、進捗の遅延が明らかになってから対処しようとすると、上記のような考慮が漏れてしまう。対策が適切でない場合は、要員を追加しても遅延状況が改善されず、さらなる要員投入や品質悪化を招く危険性もある。

本プロジェクトでは、設計工程の段階から規模増加、工数及び進捗変動を可視化し、再見積方式を含むコントロール手法を準備したことにより、このような様々な影響を考慮し

た早めの要員手配、チーム編成の見直し、要員教育等を計画的に進めることができた。図 5-10 に見直しの前後のスケジュールを示す。

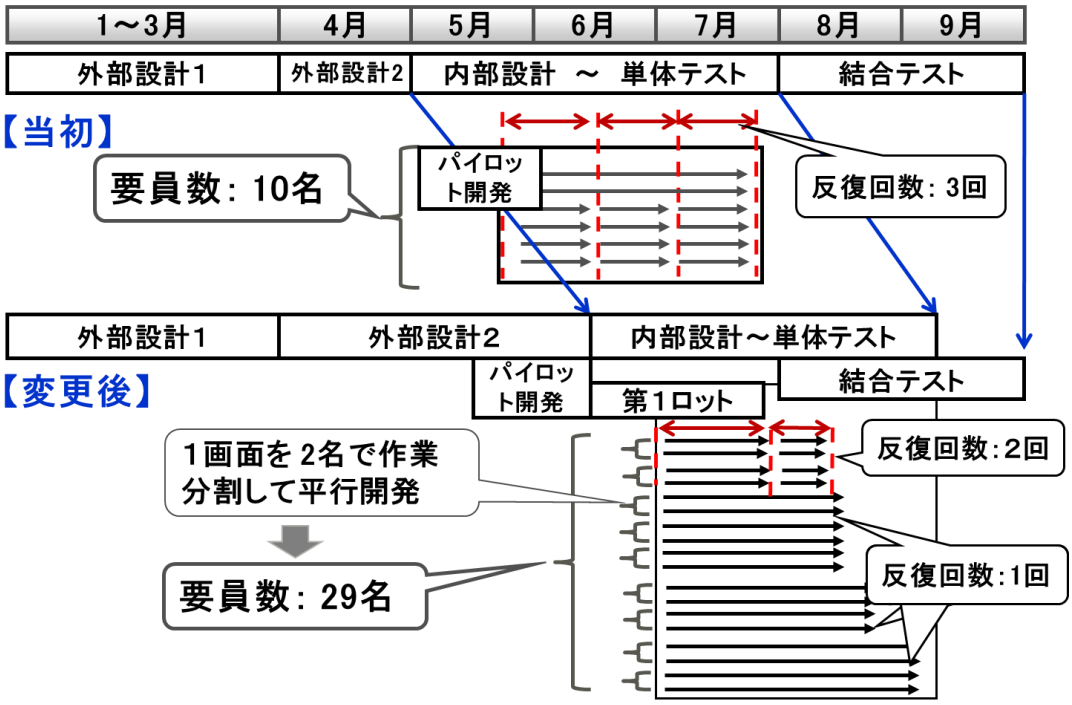


図 5-10　見直し前後のスケジュール

1 画面を 2 名で作業分割したことや、反復回数が減少したことにより、生産性は当初見積より 25%悪化したが、変動要因を考慮した再見積方式によって、レビューワの増員や追加要員への教育、資産管理要員や管理要員の増員を行い、要員数を 10 名から 29 名へと約 3 倍に増員することとなったが、以降のプロジェクトを混乱なく推進することができた。スケジュールは変更後の計画通りに進捗した。工数の再見積値と実績値の差は 5%以内であった。

なお、内部設計から単体テストまでのコストは当初の 1.8 倍の見積となった。規模増加率の 138%よりも増加率が大きい。このことから、規模に比例させて要員を投入した場合は要員数が大幅に不足した可能性があるが、変動要因を考慮した工数の再見積方式の適用によって適切な要員数を配置することができた。

本手法を適用せず、規模の増加や変動を考慮した工数見積りを行わなかった場合には、当初の 10 名のままで内部設計を進め、後工程になってからでは要員の追加が不可能となるなど、5 カ月以上のスケジュール遅延が発生した可能性がある。これに対して、本手法を適用することで事前に要員追加と計画修正を行ったことにより、見直し後のスケジュール通りにプロジェクトを進めることができた。

5.3.10. プロジェクト計画の修正及びコントロール

変動に対する再見積方式によって工数の再見積を行い、適正な要員を追加投入してプロジェクト計画を変更する。プロジェクト計画の作成は、図 5-11 に再掲する変動マネジメント手法の中の「WBS マスターシート」に、見直した後の規模、生産性、スキル、習熟度等を設定することによって、全ての機能の基準工数を一括で再計算することができるため、計画変更を行う際に、WBS の追加が発生しても、テンプレートを修正して各パラメータを見直すことによって、全ての機能の工数計画を一括作成することが可能となった。

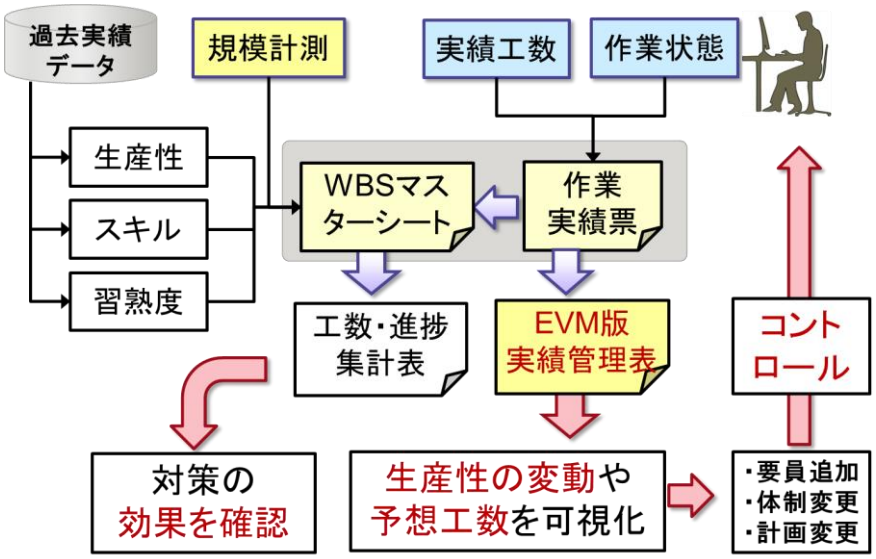


図 5-11　変動可視化手法を活用したモニタリングとコントロールのプロセス

再作成した工数計画を新たなベースラインとして、変動マネジメント手法を利用することによって、計画変更によるその後の変動の有無を監視し、コントロールによる対策の効果を確認しながらマネジメントを継続することが可能となった。

5.4.　評価

システム開発プロジェクトに対して変動マネジメント手法を適用した結果によって、提案した手法（デイリーEVM 手法）の適用効果を評価する。評価の観点は、5. 1 節の(4)で示した観点を用いる。

5.4.1. 変動マネジメント手法による効果

表 5-2 に、変動マネジメント手法による、知識継承及びマネジメントの効果について、事例としたプロジェクトの母体組織において従来から適用してきた EVM 手法（以下、既存

EVM 手法と呼称する）との比較によって示す。この既存 EVM 手法は、多くの組織やプロジェクトで行われている一般的な EVM 手法である。

表 5-2　変動マネジメント手法による知識継承及びマネジメントの効果の比較

No.	プロジェクトマネジメントのアクティビティ	既存 EVM 手法	変動マネジメント手法（デイリーEVM 手法を含む）	
			知識継承方法	知識継承とマネジメントの効果
1	WBS 作成（5.3.1 節）	プロジェク毎に個別に作成	詳細 WBS を設定した標準テンプレートの利用による熟練者の経験や知識の継承（形式知化）	・WBS の数は、既存手法の約 8 倍以上に増加したが、標準テンプレートの利用により、従来と同等の 1.5 日で全ての機能の WBS を漏れなく作成した。 ・標準テンプレートを元に作成することで、過去のプロジェクトの知識継承の効果が得られた。例えば「仕様理解」や「QA（質問）」、「テストデータの作成」、「レビューによる指摘後のドキュメント修正」などの WBS は既存の手法では含まれていなかった WBS であるが、それらの WBS の必要性を、熟練者等に確認し理解しながら計画に組み込み、知識を獲得した。 ・メンバー向けの WBS の説明書の作成に、WBS テンプレートを再利用することができた。
2	工数計画作成（5.3.1 節）	・プロジェクト毎に工数計画を個別作成し、そのノウハウは暗黙知となっていた。 ・規模と生産性だけからの工数計画であり人別	・人別の生産性や反復回数を考慮した詳細な粒度の工数計画作成 ・単位工数と規模、生産性、習熟度から自動作成（蓄積データ	・既存手法では同等の機能数の工数計画を手作業で作成するのに約 2 日の作成時間を要していた。本手法によりプロジェクトマネジメント経験が 3 年のプロジェクトマネジャーでも、規模、生産性、習熟度等を WBS マスターシートに設定するだけで工数計画を作成でき、1 日で全て機能別工数計画の作成を完了した。

		の生産性や習熟度は考慮しない。	利用と形式知化）	
3	工数計画の修正 （5.3.1 節） （5.3.10 節）	機能別、担当者別の工数計画と実績の差の補正は、初期の計画作成と同等の負荷が必要であった。	実績データを元に、単位工数や生産性を見直したテンプレートだけを修正することで全体の工数計画を一括で再作成	・既存手法では、人別や反復回数を考慮せず、人によって最大で約 70%の計画と実績での工数の乖離が発生した。 ・生産性を変更するだけで工数計画を一括で作成可能な仕組みにより、パイロット開発及び初期開発の実績データによって計画の信頼性を高めることができた。本プロジェクトでは、初期計画で 32%以上の差異が発生したが、見直し後の計画では、機能によって最大で約 10%の差異に止まった。WBS 別の最終的な工数計画と実績工数の差異は 5%以内であった。
4	工数の再見積り （5.3.9 節）	要員追加などの計画変更の際の見積りは個人の暗黙知となっていた。	習熟度や期間変更の生産性への影響や作業分割などの影響を考慮した見積り手法を形式知化	・既存手法では、要員追加等が遅れ、工数見積りが過小となり、最大で 5 カ月以上の遅延の可能性があったが、本手法により、計画どおりに推進できた。最終的な工数の計画と実績の差異は 5%以内であった。

表 5-2 に示すように、提案した変動マネジメント手法としての「デイリーEVM 手法」によって、従来はプロジェクトマネジャーの暗黙知となっていた変動に対するマネジメントのための WBS 策定、工数計画作成と計画の容易な補正が可能となり、人別の計画と工数の差異を機能別で最大で 10%に抑え、プロジェクト全体での最終的な計画と実績の差異を 5%以内とすることができた。

また、変動の定量的把握、変動の原因分析、原因に応じた適切なコントロール等の一連の知識を組織内で継承することが可能となり、経験の浅いプロジェクトマネジャーでも、定量データと過去の知識を活用することで、変動の早期察知や対処による円滑なプロジェクトのコントロールが可能となった。本手法を適用しなかった場合の想定では、規模の増加に関する影響で 38%のコスト増加の可能性があった。変動への対処が遅れ、要員追加が遅れた場合に、工数の再見積り手法の誤差と合わせると、最大で 5 カ月以上のスケジュール遅延が発生したが、提案した変動マネジメント手法によって、スケジュールは計画通り

で推進でき、工数は計画と実績の差異を 5%に抑えることができた。

なお、本手法を適用したプロジェクトは、同一の組織内の要員による、企業向けの情報システムを開発するプロジェクトであるため、過去のデータを利用でき、WBS の粒度や単位工数、習熟度などの各種の定数も、過去のデータを活用できた。このように、本手法を有効に活用するためには、組織内で実績データを蓄積してゆくことが重要となる。システムの開発方式がウォーターフォール開発モデルでなくアジャイル開発方式などに大きく変わる場合は WBS の見直しが必要になる。システムの開発方式や要員の必要スキル等が大幅に変わるような場合には、過去のデータを活用できない可能性がある。ただし、このような過去のデータが入手できない開発方式や分野のプロジェクトにおいても、本手法と同様の考え方で WBS を策定し、生産性や習熟度などのパラメータによって計画作成や変動可視化を自動化する仕組みを準備しておくことにより、先行開発工程における実績データを利用し、定数を補正することで、初期の段階で精度を高めることが可能であることを検証できた。

5.4.2. 変動把握の粒度の詳細化、精度及び運用負荷軽減の評価

表 5-3 に、変動への対応のためのマネジメントの粒度、精度及びマネジメントプロセスの運用負荷に関する本手法の効果を、既存の EVM 手法との比較によって示す。

表 5-3　変動把握のマネジメントの粒度、精度及び運用負荷の比較結果

No.	評価観点	既存 EVM 手法	変動マネジメント手法 （デイリーEVM 手法含む）
1	実績集計時間 （5.3.2 節）	5 分／日 （一人当たり）	・既存手法と同等の負荷（5 分／日（一人当たり）で、毎日の機能別、人別の変動可視化が可能となった。
2	計画値作成の負荷 （5.3.1 節）	・工程別、機能別、WBS 別に個別作成 ・WBS 数に比例した計画作成時間が必要	・25 機能の工数計画作成時間は、既存手法では 48 時間を要するが、規模や生産性から WBS 別基準工数を自動作成する本手法により 5 時間で作成できた。 ・規模や生産性の変動に応じた計画修正時は、パラメータを変更するだけで全機能の工数計画を再作成できた。 ・これによって最終的な計画と実績の工数差異は 5%以内となった。

3	実績収集負荷 （5.3.2 節）	人数、期間、勤務時間、発注金額等を組み合わせて収集するため、報告者と収集者に負荷がかかる （総工数の 1%程度）	・担当者による実績工数の入力によって詳細な粒度での進捗や工数の変動可視化を実現した。 ・担当者一人当たりでは、1 に日に 5 分以内の入力時間（総工数の 1%）でデータ入力が可能であった。
4	進捗率計上のための負荷 （5.3.2 節） （5.3.5 節）	・進捗率報告 ・成果物の量の集計 ・ヒアリング 上記作業に、毎週、半日（3 時~4 時間）を要していた。	・主に実績工数の収集により、進捗を自動計上することで、プロジェクトマネジャーやチームリーダのデータ集計作業は不要（0 時間）となり、その時間を異常の検出や問題の原因究明に振り向けることが可能となった。

表 5-3 に示すように、従来の EVM 手法では、週または月の単位でのサブシステムやチームの変動把握までしか実施できなかったが、提案したデイリーEVM 手法によって、毎日、機能別、人別の精度の高い変動把握が可能となった。また、詳細 WBS 別の実績データが蓄積されるため、異なるプロジェクトの間でのデータの活用や、データ分析の知識の継承が可能となった。また、運用負荷は従来と変わらない負荷で実施可能なことが確認できた。

以上の表 5-2 及び表 5-3 に示すように、デイリーEVM 手法の適用によって、進捗、出来高及び工数を一体とした変動を毎日把握でき、早期の異常検知や事前の対策によるプロジェクトの円滑なコントロールが可能となった。生産性や規模の見直しによる計画の修正も容易であった。この結果、仕様調整や要員追加等の対策を、設計工程における早い段階で行うことで問題の未然防止を図り、円滑なプロジェクトのコントロールに有効であることを確認できた。また、計画の修正が必要となった場合も、提案した変動アネジメント手法の適用により適切な要員追加や計画修正が可能となり、品質悪化や遅延を招くことなくプロジェクトをコントロールすることができた。

筆者が所属したシステム開発部門では、組織的なデータ蓄積に基づく定量的開発マネジメントの一貫で本手法を採用しており、7 プロジェクト以上で同様の効果を確認した。

5.4.3. 変動マネジメント手法のねらいの確認結果

プロジェクト適用によって効果を検証した点以外の、デイリーEVM 手法の機能がねらい通りに発揮された結果を表 5-4 に整理しておく。

表 5-4　既存 EVM 手法とデイリーEVM 手法の機能面の比較

No.	プロジェクトマネジメントのアクティビティ	既存 EVM 手法	変動マネジメント手法（デイリーEVM 手法含む）	
			知識継承方法	機能
1	進捗・コストの変動監視（5.3.2 節）	週単位、月単位のデータを補う属人的な確認方法	実績工数入力とルールから進捗率と出来高を自動計上（データによる補完）	・日々の機能別、人別の進捗、出来高、工数の変動を自動的に可視化することで経験不足を補うことができた。例えば変動の大きさによる重点監視によって、問題を早期に発見し、対処できた。
2	規模の変動把握（5.3.4 節）	工程完了時に、個別の方法で規模見積り実施などの属人的な方法	設計段階からの定期的、機械的な規模計測（機械的規模計測）	・規模変動の把握が遅れた場合、最大で 38%のコスト増加やスケジュール遅延の可能性があったが、外部設計工程で規模増加を把握し早期の仕様やスケジュール調整により計画通りに推進できた。
3	変動の原因分析（5.3.6 節）	・個別のヒアリングによる問題点の察知と経験や暗黙知に基づく原因分析方法	機能別、人別の詳細 WBS の実績データ活用	・WBS 別の基準工数に対する実績工数の差異等のデータから、進捗遅延などの原因となっている機能や工数、作業の特定等の分析を行うことができた。
4	変動のコントロールの知識継承（5.3.7～5.3.9 節）	変動の原因分析方法及び対策は、個人の暗黙知	対処の知識パターン化	・変動の原因に応じたコントロール方法をパターンから選択することで経験不足を補い適切な対処が可能となった。
5	実績データの蓄積・活用（5.3.6～5.3.10 節）	WBS 別の工数実績等は蓄積されにくい	詳細 WBS 別の工数実績と生産性、習熟度、見積り式の係数等のデータを蓄積	・当該プロジェクトの進行中に生産性や係数を見直し、計画の精度を向上させることができた。（最終的な工数の計画と実績の差異は 5%であった。） ・プロジェクト終了時点で、初期の計画時や変更後のデータが蓄積され、組織としてのデータを活用したマネジメント手法への活

				用に適用することが可能となった。

5.4.4. 定量的マネジメントのためのデータ蓄積の必要性

システム開発プロジェクトで発生する規模増加や様々な要因による計画と実績の差異に対して、本研究で示した方法により、適時に精度の高い工数見積と早期の仕様調整や計画修正が可能となり、後工程での問題発生を未然に防止できる。

適用事例では、規模分布と期間に関連する反復回数と生産性の関係や、作業分割に伴う要員増加などの影響を工数の再見積や要員の再配置に活用した。提案した手法を有効に活用するためには、これらの各種実績データを組織内で継続して蓄積することが重要である。

6. プロジェクトマネジメントの知識構造モデル

プロジェクトマネジャーの経験や暗黙知の共有、継承を促進するための方法の一つとして、第 4 章及び第 5 章において、プロジェクトの変動マネジメントに関する過去の知識をパターン化、システム化することによって知識を形式知化し、活用しやすくする方法を示した。

本章では、プロジェクトの変動マネジメントに関する知識の構造を一般化することによって、知識の形式知化を行う際の領域の特定や、形式知化やシステム化の具体的な方法の検討に活用可能な知識構造モデルを提案する。プロジェクトマネジメントに必要な知識を構成する知識要素の種類や、各知識要素の関係等を明らかにすることにより、継承すべき知識の特定や知識の属性に応じた形式知化やシステム化を進める際の枠組みとすることを目的とする。

6.1. プロジェクトマネジメントの知識の体系化の方法について

各組織においてプロジェクトマネジメントの知識継承の取り組みが行われているが、施策の妥当性や効果が必ずしも明らかではないケースもある。これは、プロジェクトマネジャーに必要な知識が多岐に渡り、知識の多くが個人の暗黙知となっていることに加え、具体的にどの範囲の知識を継承すべきかを明確化しにくいことや、知識の特性に適した知識継承の方法が確立されていないことが背景にあると考える。暗黙知となっているプロジェクトマネジャーの知識には、形式知化が容易なものと難しいものがある。第 4 章で提案した変動マネジメント手法を例にとると、計画策定時の WBS の策定や変動監視プロセスの準備、EVM の指標を用いた分析方法等については、様々な手法の研究や事例の成果が発表されており、比較的形式知化が容易である。進捗やコストのモニタリングについては、マネジメント手法を標準化し、実績データを蓄積することで、システムによる代替も可能となる。変動に対するコントロール方法や工数の再見積りの方法等は、第 4 章で提案したように、変動の原因別の対処方法をパターン化することによって形式知化することができる。他方で、リスクの予測やステークホルダマネジメント等においては、経験に基づく判断が求められ、暗黙知の占める割合が多い。このような暗黙知は、主に Nonaka and Takeuchi (1995) の SECI モデルにおける「共同化 (Socialization) 」 のための場づくりや体験型の教育などを中心として継承してゆくことが効果的である。

このように、継承すべき知識の範囲を明確にし、その種類や特性に応じて適切な知識の形式知化や継承方法を選択することができれば、効果的な知識の蓄積や活用が可能となると考えられる。そこで本章では、プロジェクトマネジャーの様々な役割や必要な知識を、マネジメントプロセスと関連付けて体系化した、知識構造モデルを提案する。合わせて、形式知化やシステム化が容易な機能と、暗黙知の占める割合が多い知識等に分類する方法を提案する。これにより、第 4 章で提案した変動マネジメントに関する領域以外に含まれ

る知識も含めて、より広範囲でのプロジェクトマネジメントのアクティビティを形式知化またはシステム化するための具体的な方法や、知識の蓄積方法等を検討する上での枠組みとして活用可能なものとすることを目的とする。

第 4 章では、プロジェクトの変動の可視化や異常の察知をシステム化し、コントロールに関する知識をパターン化によって形式知化することによって、プロジェクトの変動に関するプロジェクトマネジメントの知識の継承に役立てることができることを示した。このような、プロジェクトマネジメントの知識活用や、システム活用によるマネジメントの支援等に関連した研究として、近年、人工知能（以下、AI と略す）に関する技術をプロジェクトマネジメントに活用する事例が発表されている（岡田 2017, 河村 2018, 神林 2018）。これらの研究の成果から、例えば、リスクの予測や異常の検知等は、プロジェクトの特性や状況に関する様々なデータを元にした機械学習などを活用することで、プロジェクトマネジメントにおける知識や判断を補完できる可能性があることが示されている。こうした AI 技術の活用は、プロジェクトマネメントの知識をシステム化によって補う方法の一つと位置付けられる。以下で提案する知識構造モデルの活用方法の一つとして、プロジェクトマネジメントの知識を、システム化や AI 技術によって代替または補完できる可能性の有無によって分類する方法について考察する。これにより、システム化や AI 技術適用によって知識を補完する方法の検討に、知識構造モデルを活用する方策の例を示す。

6.2. プロジェクトマネジメントに関する知識の体系化

6.2.1. プロジェクトマネジメントの知識体系と知識継承について

プロジェクトマネジメントの知識体系としての PMBOK ガイドについては、第 3 章の 3.10.1 で取り上げたように、Horner and Yong (2006) が「PMBOK ガイドは、形式知や宣言的な知識に偏っており、暗黙知や理由（Why）に関する知識にはあまり注意が払われていない」と分析している。この分析は、当時の PMBOK 第 4 版を対象としたものであるが、基本的な記述内容は第 6 版でも同様である。つまり、PMBOK ガイドの記述内容は、実施すべきプロセスや利用可能な技法等が中心であり、プロジェクトにおいて発生する問題に対して、いかなる理由によりどのように対処すれば良いか等は示されていないと言える。プロジェクトマネジャーは、経験に基づく暗黙知を含む知識をプロジェクトの状況に応じて駆使する必要があるため、プロジェクトマネジメントに必要な具体的な知識を洗い出し、形式知化やシステム化を行う上では、PMBOK ガイドに記述されている内容だけでは不十分である。

本章では、プロジェクトの変動マネジメントに関する知識の要素要素を洗い出し、構造化した知識構造モデルを提案する。この知識構造モデルを元にして、形式知と暗黙知の観点での知識の分類方法を示す。

この際、ソフトウェア開発においてプロジェクトマネジャーが行う対処と管理要素や知

識の間の動的な関連性に着目する。ここで、第 4 章で示した図 4-2 の知識関連モデルを図 6-1 に再掲する。

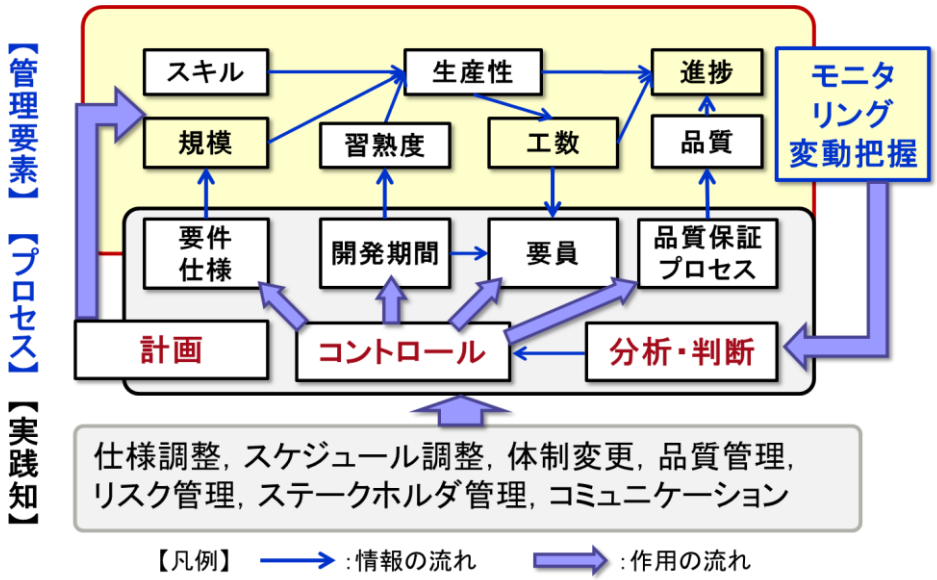


図 6-1　変動マネジメントの知識関連図（再掲）

6.2.2. 知識構造モデル

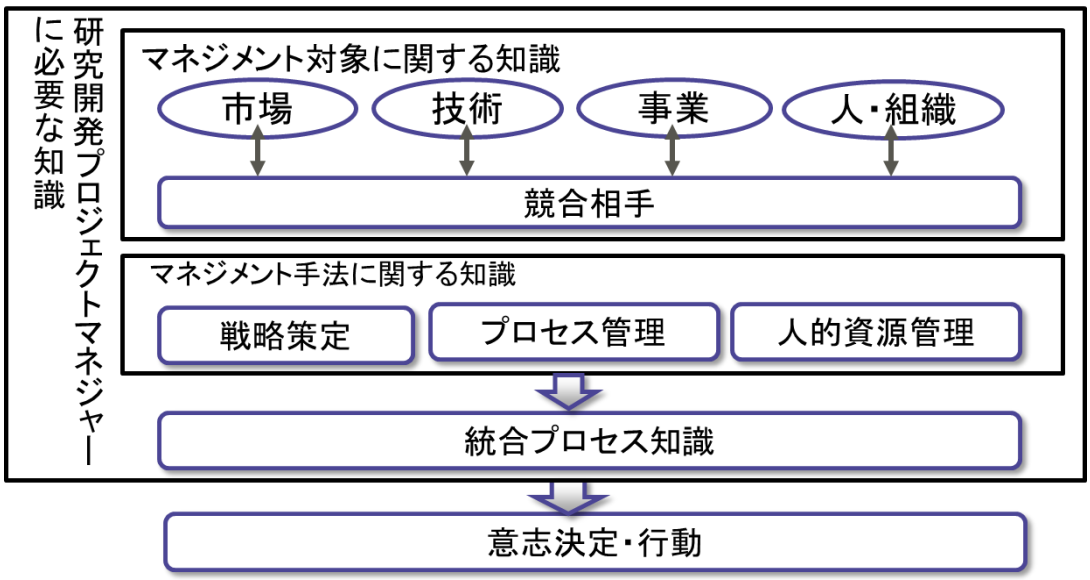


図 6-2　研究開発プロジェクトマネジャーに必要な知識

（内平（2010）図 4.1 より）

内平（2010）は、研究開発プロジェクトにおける知識継承のための知識構造化モデルに基づいた知識継承方法を提案している。この中で、図 6-2 に示す研究開発プロジェクトマネジャーに必要な知識の構造が示されている。

図 6-2 を参考に、図 6-1 の知識関連図を再構成し、詳細化した知識構造モデルを図 6-3 に示す。図 6-1 で「実践知」に分類した知識要素のうち、変動マネジメント手法におけるコントロールの知識のパターンを、対象に関する管理要素と関係づけた。図 6-2 と図 6-3 の対応関係を示すと、図 6-2 の「マネジメント手法に関する知識」が図 6-3 の「管理要素」と「プロセス」に相当し、「統合プロセス知識」が図 6-3 の「実践知」に対応する。ここで、実線の矢印は情報の流れを示し、太枠の矢印は作用の流れを示している。

本モデルでは、過去の知識の活用や形式知化が行ない易くなるように、「変動管理」や「コントロール」のプロセスと関連付けて、知識要素、管理要素及び対処方法の間の関係を示している点が特徴である。さらに、知識の特性によって以下の 5 つの知識レイヤーに分類し、知識間の関係を表現している。

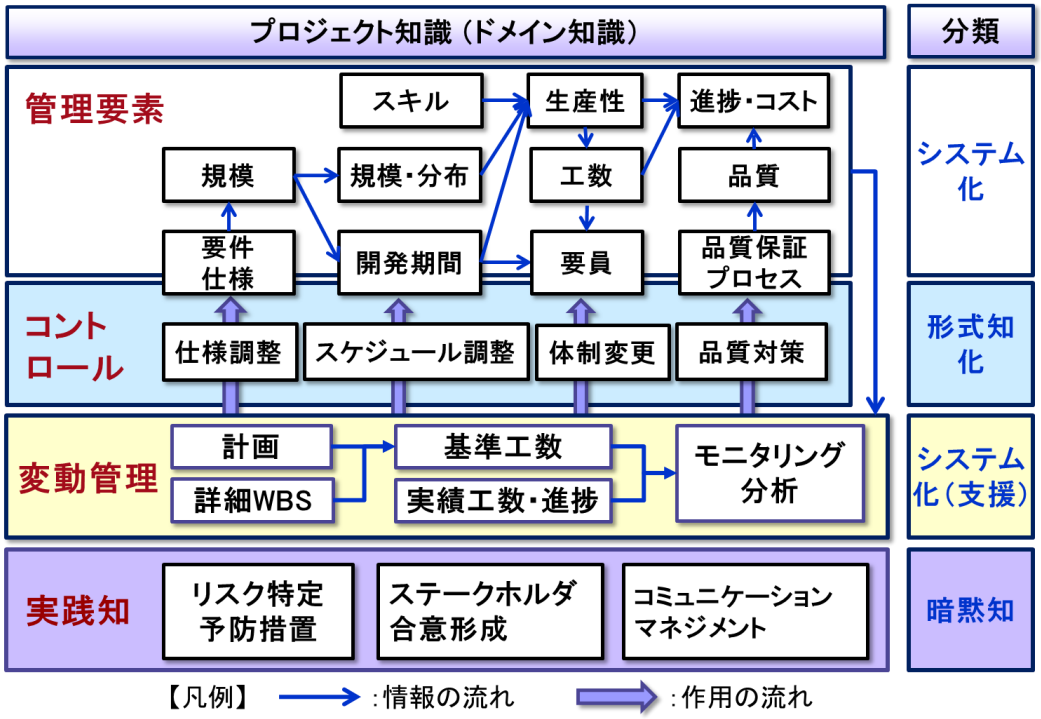


図 6-3　プロジェクトマネジメントの知識構造モデル

上記の知識構造モデルは、ソフトウェア開発プロジェクトの変動のコントロールを対象としたものであるが、他分野への応用も可能と考える。以下に、図 6-3 の知識構造モデルの各知識階層について説明する。

(1)　プロジェクト知識

プロジェクト自体のドメインに関する知識や開発手法、組織、構成員等の知識である。

図 6-3 では詳細は省略している。

(2) 管理要素

規模、期間、工数、要員、スキルなどのリソースに関する知識や、進捗、生産性、品質などのプロジェクトの状態を表すメトリクス等のプロジェクトマネジメントに利用する各種メトリクスに関する知識、及び要素間の関係性に関する知識から成る。これらの要素は、変動管理プロセスやコントロールプロセスから参照される。

管理要素間の関係の一例を挙げると、例えば、工数は、規模と生産性から求められるが、スキルや開発期間によって生産性が異なるという関係性により、工数に影響が及ぶことを示している。また、進捗やコストは生産性や品質状況に影響されることなども表している。このような関係性に関する知識を「管理要素」として分類したものである。

なお、図の管理要素に示した要素は主要なものを示しており、実際には、成果物やプロセスに関する各種の管理要素と、それぞれの管理要素を表す各種のメトリクスガ含まれる。また、要素間の関係性についても、図には主な関係のみ示している。

(3) コントロール

変動管理プロセスのアウトプットを元に行う各種の対策から成る。問題の発生原因に応じた仕様調整やスケジュール変更、体制変更、品質対策などの主要な対策をパターン化しておくことで、プロジェクトマネジャーへの行動指針の提示や知識の効果的な共有が可能となる。例えば、仕様追加等によって規模が増加した場合は、プロジェクトオーナーなどのステークホルダとの仕様調整を行うことで、規模の増加を抑制する。進捗遅延やコスト増加に対しては、その原因に応じて、要件・仕様、開発期間、要員、品質保障プロセス等に変更を加えることで、体制変更やスケジュール調整などを行う。こうしたコントロールのための知識の共有や継承を行う場合に、この知識構造モデルを活用することができる。

具体的なコントロールのパターンは、図 4-20　変動可視化によるコントロールのパターン」として、この知識構造モデルとは別に記述して利用する。

(4) 変動管理

規模、工数、進捗等の変動を可視化するための管理プロセスであり、工数計画、実績収集及び進捗計上方法等から構成される。第 3 章、第 4 章お呼び第 5 章で示したように、システム化することで、計画と実績の差異や変動のモニタリングを少ない運用負荷で行うことが可能な知識レイヤーである。

(5) 実践知

リスクの特定、ステークホルダマネジメント、コミュニケーションマネジメントなどに必要な、実践経験から得られる暗黙知に位置付けられる知識である。ここでの実践知は、Nonaka & Takeuchi (1995) が定義している、「絶えず変化する状況や文脈の中で、最善の判断ができる実践的な知性」という意味で用いている。

6.2.3. 知識レイヤー間の動的関係モデル

図 6-3 の知識構造モデルおける各知識レイヤー間の動的関係を表したモデルを図 6-4 に示す。ここで、動的関係とは、プロジェクトマネジメントにおける監視（変動の管理）と制御（コントロール）を実行する際に、プロセスごとに、どのレイヤーの知識を組み合わせて活用するのかという、知識レイヤーとマネジメントプロセスの動的な組合せ関係を表す意味で用いている。プロジェクトマネジメント活動を、制御対処としてのプロジェクトに対する活動と捉えてモデル化したものであり、以下に示すように、状況に応じて必要となる知識を動的に組み合わせ、監視と制御を繰り返し実行することによってプロジェクトの変動が少なくなるようにマネジメントするという意味合いで、「動的関係」と表現した。図 6-3 の知識構造モデルは、主として知識要素の間の静的な因果関係を示したものであるが、図 6-4 の動的関係モデルは、マネジメントのプロセスの視点で示したもので、図 6-3 を補完するものである。

これらのモデルによって、各知識レイヤーの知識をプロジェクトマネジャーがマネジメント活動の中のどの局面で活用するのかが明確になる。

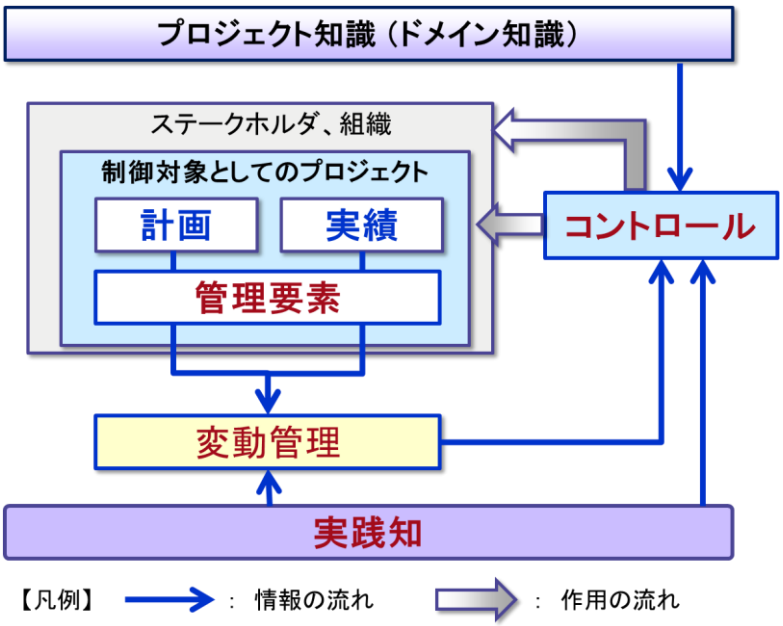


図 6-4　知識レイヤー間の動的関係モデル

図 6-1 のプロジェクトマネジメントの概念図で示したように、プロジェクトマネジャーの行動を、「プロジェクトの変動を把握して計画と実績の差異を少なくなるようにコントロールすることである」と捉える。まず、計画と実績の差異を、管理要素を用いて定量化して把握する。その差異の変化を継続的に把握して「変動管理」を行う。変動管理の結果を用いて差異が少なくなるようにプロジェクトをコントロールし、その効果として変動が少

なくなったか等について、変動管理によって引き続き監視してゆく。この際に、実践知を活用してステークホルダマネジメントなどと合わせたコントロールを行う。その後の変動状況の監視を継続して行うことによって、コントロールの効果を確認する。このサイクルを繰り返すことでプロジェクトを計画通りにコントロールする。なお、このモデルにおいて、ステークホルダや組織はプロジェクトを取り巻く環境として定義している。図 6-3 の知識構造モデルと図 6-4 の知識間の動的関係モデルによって、プロジェクトマネジメントに必要な知識を、どのような目的でいかに活用するかという、Why や How の観点で知識の共有や継承に役立てることができる。この点は、PMBOK などのプロジェクトマネジメントの知識体系には不足していた点であり、本モデルを活用して知識体系を整理することで、これらの点が改善されると期待される。

6.3. 知識構造モデルの変動マネジメントと知識共有への活用

4 章に示した一連の変動マネジメントプロセスを、図 6-3 の知識構造モデルに沿って再構成して示すことによって、提案した知識構造モデルが、プロジェクトマネジメントの知識の体系化や、知識継承のための枠組みとして利用可能であることを示す。

6.3.1. 変動マネジメントの手法への適用

4 章のプロジェクトの変動マネジメント手法に関連する知識を、図 6-3 の知識構造モデルと図 6-4 の動的関係モデルに沿ってモデル化したものを図 6-5 に示す。

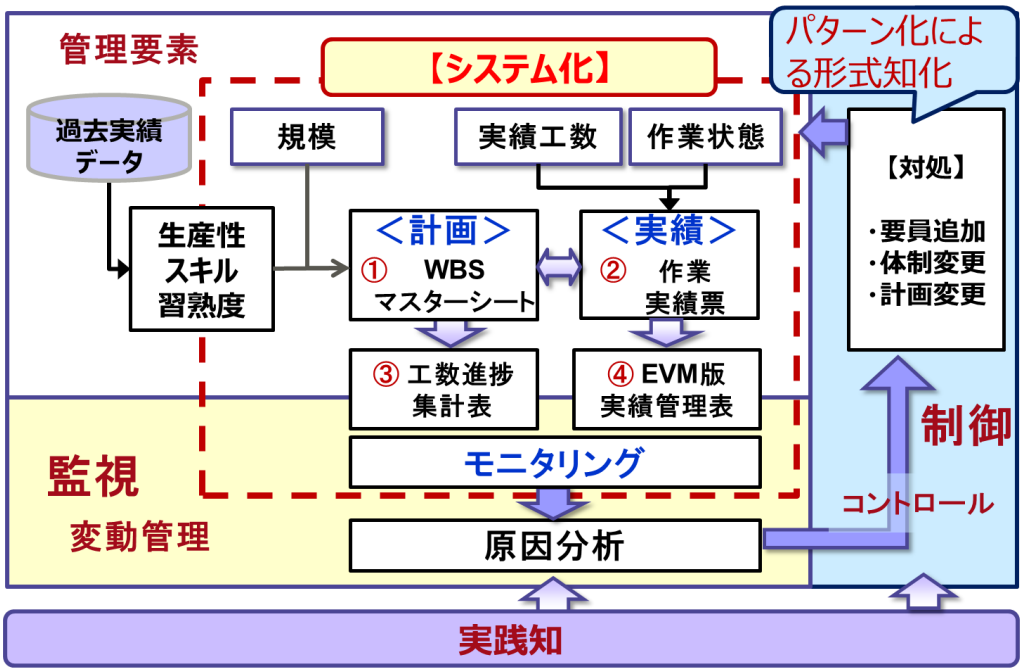


図 6-5　変動マネジメント手法と知識構造モデルの対応関係

図 6-5 に、変動マネジメント手法と、知識構造モデル（図 6-3）の対応関係を示す。図 6-5 の内側の破線で囲んだ部分がシステム化した範囲であり、知識構造モデル（図 6-3）の「管理要素」をインプット情報とするモニタリングのプロセスに対応する。システム化した領域の外側の原因分析、判断及びコントロールの一連のプロセスが知識構造モデル（図 6-3）の「コントロール」の層に対応しており、このプロセスを標準化することでシステムと合わせた形式知化を図ることができる。

6.3.2. 変動マネジメントの知識活用における知識構造モデルの適用

ここで、変動マネジメント手法を活用してマネジメントを行う場合の知識を活用する上での知識構造モデルの適用例を示す。ここでは、変動マネジメント手法の中の、EVM 版実績管理表を利用したマネジメントの場合を例にとって説明する。

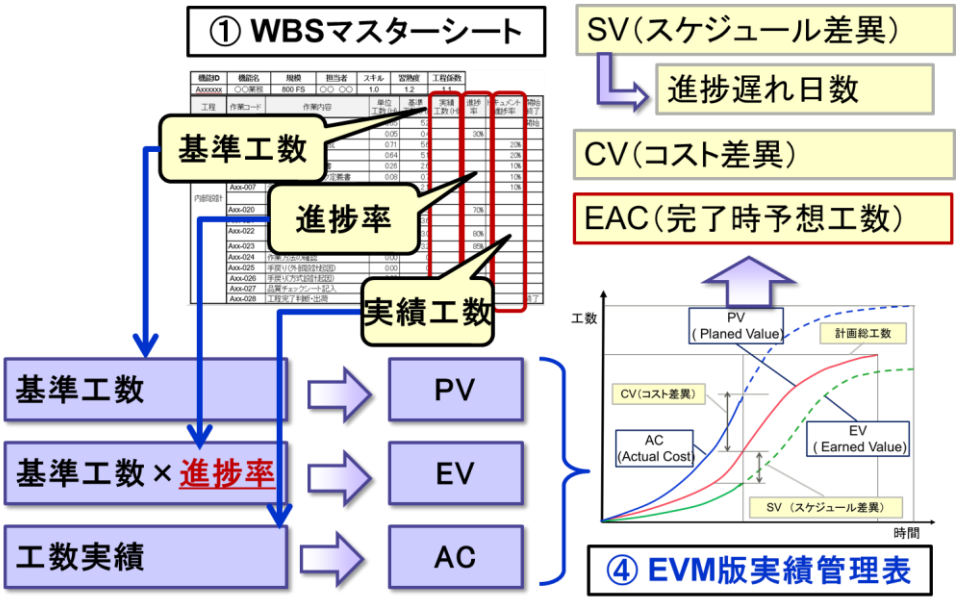


図 6-6 「EVM 版実績管理表」と管理要素の関係

「EVM 版実績管理表」を構成する管理要素と EVM の各指標の間の関係を図 6-6 に示す。「EVM 版実績管理表」によって、スケジュール、コストや生産性の変動傾向を継続的に把握することができる。例えば、SV（スケジュール差異）から、進捗遅れを日数で直感的に把握でき、完了時の予想総コストとしての EAC (Estimate At Completion) が毎日更新される。規模の変動状況については、第 6 章の規模変動可視化手法を適用することで把握できる。

この変動可視化手法を活用してプロジェクトマネジメントを行う際の、知識構造モデルの活用例を示す。一例として、内部設計工程での活用事例を示す。図 6-6 の EVM 版実績管

理表から、PV（計画値）と EV（出来高）の差は少なく、進捗は予定どおりであるが、AC（実績工数）が増加しておりコストが計画よりも拡大傾向である場合を取り上げる。

図 6-3 の知識構造モデルを参照すると、工数や生産性に影響する要因として規模、要員スキルや習熟度、開発期間等が挙げられることがわかる。そこで、これらの要因のどこに原因があるかを分析する。例えば、「WBS マスターシート」に集計された実績工数を確認すると、「仕様理解」の実績工数が基準値よりも増加していた場合、設計書の品質に原因があると推測でき、特定の設計書のレビュー強化などの適切な対処が可能となる。このような変動の原因に応じた対応方法を知識構造モデルと対応させてパターン化しておくことで、経験の浅いプロジェクトマネジャー向けのガイドとして活用することもできる。このように、図 6-3 の知識構造モデルは標準的な知識体系との対応も行いやすく、形式知化やシステム化が容易な知識領域の分類に活用できることと、プロセスと関連付けて知識を構造化しているため、知識の蓄積や活用を進めやすいことが特徴である。

6.4. 知識構造モデルの特徴

前述のように、図 6-3 の知識構造モデル及び図 6-4 の動的関係モデルの特徴は、PMBOK や ISO21500 などのように知識領域やプロセスをサブジェクトグループやプロセスの種類で分類するだけではなく、マネジメントプロセスと管理要素や各知識と対処方法の間の動的な関係を構造化し、モデル化した点である。これにより、プロジェクトマネジメントの具体的なプロセスの策定や知識継承、及びデータの蓄積と活用等に利用できることを狙いとしたものである。

例えば、マネジメントの各プロセスにおいて「どのような情報（管理要素）を元に、何を（対象資源）どうコントロールすれば良いか（Why、How）」といった観点で知識を活用しやすく、継承すべき知識も明確になる。

次に、図 6-3 の右側に、「形式知化」、「システム化」、「システム化（支援）」及び「暗黙知」という分類を示している。これは、「管理要素」、「コントロール」、「変動管理」、「実践知」の各レイヤーを、形式知化やシステム化が比較的容易な領域と、形式知化が難しく暗黙知の占める部分が多い領域に分類したものである 。ここでの形式知と暗黙知は、Milton (2005) のプロジェクトマネジメントにおける知識分類の定義に従い、「人の頭の中にある体験に基づく知識」を「暗黙知」とし、「記録された知識」を「形式知」として使用する。これらの知識レイヤーごとの分類は相対的なものであり、形式知化が可能な知識レイヤーにも、結果の分析や判断などを行う際には暗黙知が必要とされることがある。

この知識構造モデルにより、計画策定やモニタリング、コントロールの各場面においていかなる情報を元に、どのような対処を行えば良いかといった観点で知識を整理することで、知識の共有が容易となり、継承すべき知識が明確になる。また、各知識レイヤーの特性に応じて、各組織内で効果的な知識継承を行うことが可能となる。例えば、形式知化や

システム化を行いやすい知識レイヤーについては、プロセスや知識の間の関係性と結びつけ、利用しやすい形で知識を整理することができる。暗黙知の占める部分が多い知識レイヤーについては、SECI モデルにおける「共同化」を進める際の方法の検討などに役立てることが可能である。以上のように知識を構造化することにより、組織的な知識の蓄積や活用を進める際の枠組みとして活用することができる。

6.4.1. 標準知識体系との関係

次に、図 6-3 の知識構造モデルにおける形式知化やシステム化、暗黙知等の分類と、標準知識体系の一つである ISO21500 のプロセス関連テーブルとの対応関係を図 6-7 に示す。

以下に、知識構造モデルの各知識レイヤーと ISO21500 のプロセスとの間の対応関係を例示する。

(1) プロジェクト知識レイヤーについて

知識構造モデルにおけるドメイン別の知識を意味している。この知識レイヤーについては、ISO21500 では明示されていない。

(2) 管理要素知識レイヤーについて

この知識レイヤーは、プロジェクトのモニタリングとコントロールを行う際の可視化や分析の対象となるプロジェクトの特性やメトリクスに対応するものであり、プロセスを中心に分類されている ISO21500 においては明示されていない。ただし、管理要素との対応関係が強いプロセスとしては、スコープ、資源、タイム、コスト、品質の各サブジェクトグループが挙げられる。

また、ISO21500 ではプロセスの間や、プロセスのインプットとアウトプットの関係は示されているが、知識構造モデルで示している管理要素間やコントロールレイヤとの間の関連性については明示されていない。

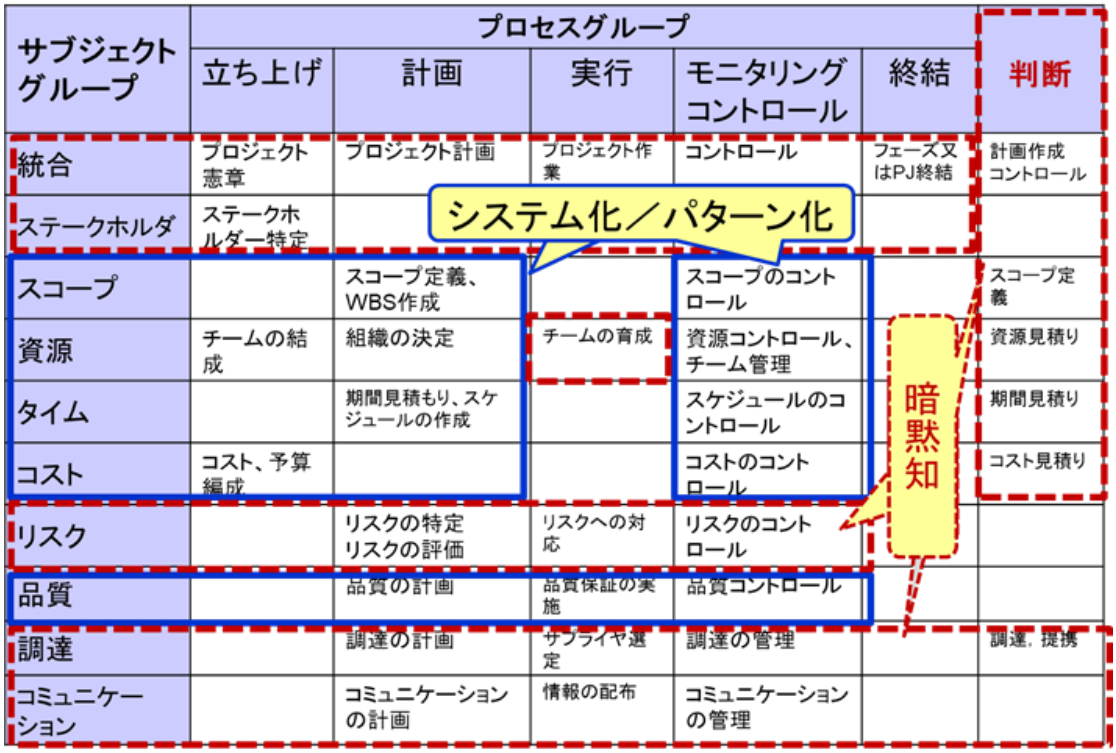


図 6-7　ISO21500 のプロセスと知識構造モデルの関係

(3)　コントロール知識レイヤーについて

知識構造モデルのコントロール知識レイヤーについては、直接には ISO21500 のコントロールプロセスに対応する。ただし、知識構造モデルにおいては、問題の発生原因に応じた仕様調整やスケジュール変更、体制変更、品質対策などの主要な対策についてもパターン化してモデルに記述しておく点が ISO21500 や PMBOK には含まれていない点である。

(4)　変動管理知識レイヤーについて

この知識レイヤーについても、ISO21500 には該当するプロセスがない。ただし、変動管理に含まれるモニタリングは、コントロールと一体として行われるものである。

また、変動管理知識レイヤーには、変動管理のベースラインとしての計画作成や WBS 作成、実績データ収集の準備及び蓄積なども含めているため、立ち上げ、計画、実行、終結のプロセスとも関係がある。

(5)　実践知レイヤーについて

この知識レイヤーは、経験によって習得される実践的な知識であり、上記(1)から(4)の知識の背景知識として存在するものである。ISO21500 における統合、ステークホルダ、リスク、調達、コミュニケーションマネジメントの各サブジェクトグループとの関係が深い。

また、図 6-7 では、ISO21500 のプロセスの中で、図 6-3 の知識構造モデルで示したシステム化や形式知化が可能なレイヤーに対応する部分を枠で囲んで示している。実線で囲んだスコープ、資源、タイム、コスト、品質に関するプロセスは比較的、標準化やシステム化が容易な部分である。その他の点線で囲んだ部分は、暗黙知が占める割合が多い。

このように、図 6-3 の知識構造モデルは、ISO21500 や PMBOK の知識分類要素を包含したものである。ISO21500 や PMBOK では、プロセスマネジメントのプロセスを横軸（フェーズ）と縦軸（プロセスの種類）で分類したものであるため、各知識の要素間の関係性が表現されておらず、知識要素が断片化した形になっている。このため、知識の網羅性を重視した体系となっているが、実際のプロジェクトマネジメントにおいては、例えば、コストが変動した場合には、コストに影響を及ぼす可能性のあるソフトウェア規模、人別のスキルや生産性、開発期間などの知識や、影響要因相互の関係性の知識を活用しながらコントロールを行うため、断片的な知識の集合では活用しにくい。これに対して、本件研究で提案した図 6-3 の知識構造モデルは、各知識が、プロジェクトの監視や制御におけるどのプロセスに対応したものであるかと、知識要素間の関係が明示されており、知識の形式知化やマネジメントプロセスでの知識活用の観点で利用しやすいモデルとしたものである。

6.4.2. 知識構造モデルに対応するアクティビティの例

次に、形式知化やシステム化の容易性の観点で知識レイヤーを分類可能となっている点の活用効果について以下に示す。初めに、図 6-3 の知識構造モデルと知識分類に基づき、形式知が容易な知識レイヤーと暗黙知の占める部分が多い知識レイヤーの具体的なアクティビティとその分類について例を示す。

6.4.2.1 形式知化やシステム化が比較的容易なアクティビティの具体例

以下に、図 6-3 における「管理要素」、「変動管理（モニタリングを含む）」及び「コントロール」のそれぞれの知識レイヤー別に、形式知化やシステム化が可能な具体的なプロジェクトマネジメントのアクティビティを例示する。

(1) 管理要素（規模、コスト、品質、期間等）に関するアクティビティ
  (a) 計画工程における規模見積り、工数見積り、WBS 作成
  (b) 品質計画作成
(2) 変動管理（モニタリング）に関するアクティビティ
  (a) 進捗・コストの変動の把握、可視化
  (b) 品質状況の可視化
(3) コントロールに関するアクティビティ
  (a) 対処方法のパターン化

プロジェクトのコントロールでは、変動の原因分析に基づき、対処方法を決める必要があるため、プロジェクトマネジャーによる判断や決断が必要となる。この知識レイヤーにおいては、発生した変動に応じた仕様調整、スケジュール調整、体制変更、品質対策などの対応方法を、変動の種類や原因に応じてパターン化することで形式知化できる部分を拡大することができると考える。

(b) 工数や要員数の再見積りの方法の形式知化

規模や開発期間などが変動した場合に、計画の修正に必要となる工数や要員数を見積り直すことが必要となる。このアクティビティはプロジェクトマネジャーの経験に基づいて行われることが多いが、規模や期間の変動の影響に関するデータを蓄積することによって形式知化することが可能である。

以上のアクティビティのうち、(1)の(a)、(2)の(a)は、大島・丸山（2017）によってシステム化の方法が提案されており、(3)の(b)は、大島・丸山（2016b）で形式知の方法が提案されている。

6.4.2.2　暗黙知の占める部分が多いアクティビティの具体例

6.4.1.1 で示した知識レイヤー以外は、形式知化が難しく暗黙知の占める部分が大きい領域である。特に、プロジェクトに影響を及ぼす外部環境、組織文化や人に関連する部分のマネジメントなどは、様々な活動や事象の積み重ねによって共通の認識を獲得するものであり、言語化や形式知化することが難しい。以下に、PMBOK や ISO21500 の知識領域やプロセスの中から、暗黙知の占める部分が多いアクティビティの具体例を示す。

(1) プロジェクト・ライフサイクルと組織

(a) プロジェクトマネジメントに影響を及ぼす企業の理念や組織文化の理解

(b) 国や地域の文化、習慣の理解

(2) 人的資源マネジメント

(a) 個人のスキルや特性などを考慮したプロジェクトチーム編成

(b) パフォーマンスの良いチームとしてメンバーを育成するためのチームビルディング

(3) ステークホルダマネジメント

ステークホルダの特定と、ステークホルダの特性に応じた合意形成

(4) リスクマネジメント

リスクの特定や、事前の対策、リスクの予兆検知、リスク顕在化時の判断

(5) コミュニケーションマネジメント

組織や顧客に関する文化や風土に関する認識

ここで、ISO 21500 の 3.3.38　コミュニケーション計画書の説明で、「多様な文化などの要因、及び組織的要因は、コミュニケーション要求事項に著しい影響を与えることがある。」

と記述されているように、(5)(a)のプロジェクトマネジメントを取り巻く組織や顧客に関する文化や風土、(5)のステークホルダに関する認識や(4)のリスクの特定等については、経験によって習得する部分が多く、プロジェクトマネジャーの実践知と位置付けられる部分であり、暗黙知が占める割合が多い領域であると考えられる。

なお、各知識エリアに共通する人間関係の知識として、心理、認知バイアス、個人特性などの知識・経験や、人材育成、リーダーシップ等が挙げられるが、これらはプロジェクトマネジメントの知識構造モデルでは明示していない。

6.5. プロジェクトマネジメントアクティビティの知識分類

6.5.1. プロジェクトマネジメントの知識の継承方法による分類

前節での考察を元に、図 6-3「知識構造モデル」に示されている知識構造に対応した主なプロジェクトマネジメントのアクティビティを抽出したものを図 6-8 に示す。ここでは、ISO21500 のプロセスグループに対応させて示している。抽出したアクティビティは、図 6-3 の知識構造モデルにおける変動管理、コントロール及び実践知の各知識レイヤーに関するアクティビティを中心としたものである。

図 6-8 では、抽出したアクティビティを、形式知化が容易なものと暗黙知の占める部分が多いものに分類している。「知識分類」の欄は、形式知化が容易なアクティビティについて形式知の欄に「○」を付し、形式知化が難しく暗黙知の占める割合が多いアクティビティには暗黙知の欄に「○」を付している。なお、形式知化の欄に「○」を付しているアクティビティの中で、暗黙知の欄に「△」を付している箇所があるが、これは、形式知化が容易なアクティビティについても、全てを形式知化できるとは限らず暗黙知の要素が含まれることを示している。以上の分類結果に対して、形式知の欄が「○」のアクティビティを実線で囲み、暗黙知の欄が「○」のアクティビティを点線で囲んで示す。

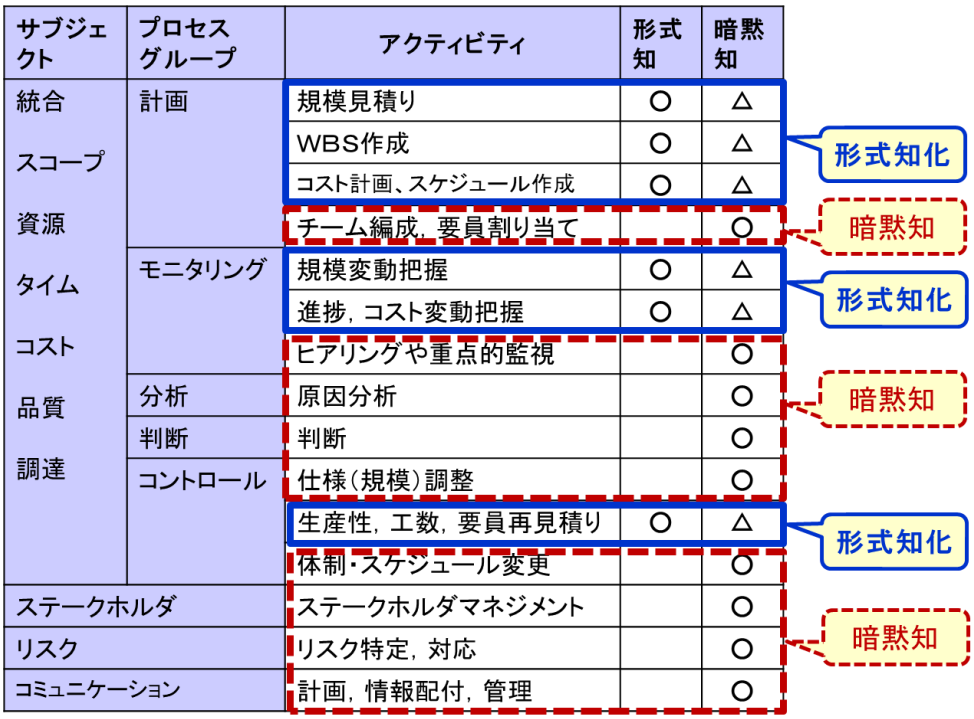


図 6-8　マネジメントアクティビティの知識分類

ただし、この形式知と暗黙知の分類は、相対的な分類である。また、暗黙知に分類しているアクティビティでも、形式知化が可能な部分があると考えられる。

6.5.2. プロジェクトマネジメントへの AI の適用可能性による知識分類モデル

前項までの知識の分類と、利用可能なデータの有無等に基づいて、プロジェクトマネジメントの知識を、形式知化、システム化及び AI 技術適用等の、どのような方法を知識継承に利用可能かという観点で分類、整理する。

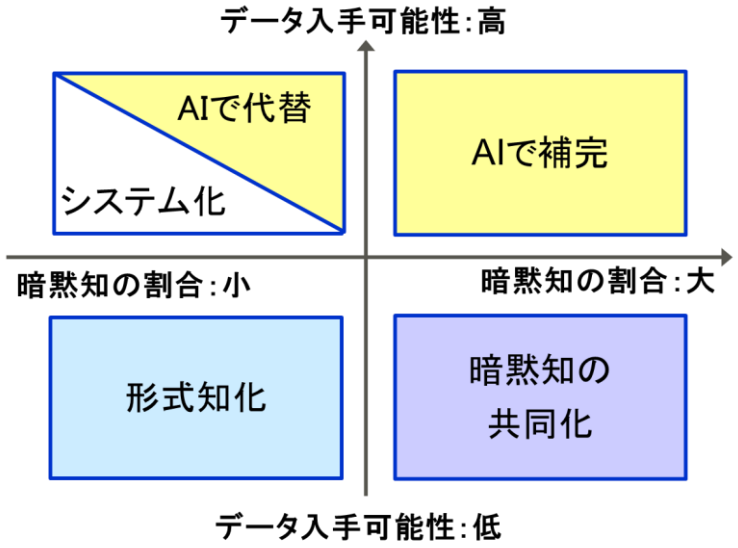


図 6-9　知識継承の方法や AI の適用可否による知識の分類方法

ここでは、「システム化」という用語は、「AI 技術」を利用せずにシステム化を実現できることという意味で使用する。AI 技術の定義や範囲については、次章の 6.6.1 で示す。

図 6-9 に、プロジェクトマネジメントの知識やアクティビティについて、システム化や AI 適用等のいずれの方法が適用可能かを、データ入手の可能性や暗黙知の占める度合いによって、分類するための考え方を示す。図 6-9 は、横軸に暗黙知の占める度合いを取り、縦軸には分析や学習のインプットとなるデータの入手可能性を取ることによって、対象となるプロジェクトマネジメントのアクティビティがどの領域に位置付けられるかを分類するものである。この分類軸を利用すると、例えば、暗黙知の割合が少なく、データ入手の可能性が高いアクティビティはシステム化が可能な領域である。暗黙知の割合が多く、データ収集が比較的容易なアクティビティについては、システム化は難しいが、AI 適用によって知識の補完ができる可能性のある領域と言える。なお、暗黙知の割合が少なくデータ入手が容易な領域はシステム化が可能であるが、その中でも暗黙知の割合が多くなるほど、従来型のシステム化には限界が生じてくる。このような領域については、データの入手可能性が高ければ AI の適用によって代替できる可能性があるということを表している。

次に、図 6-9 の知識継承方法による分類を元に、具体的なプロジェクトマネジメントのアクティビティの分類例を図示す。ここでは、ソフトウェア開発プロジェクトにおけるプロジェクトマネジメントのアクティビティの分類例を示す。

サブジェクト	プロセスグループ	プロセス，アクティビティ	形式知	暗黙知	利用データ	システム化	AIで代替	AIで補完
統合 スコープ 資源 タイム コスト 品質 調達	計画	規模見積り	○	△	○	△	○	
		WBS作成	○	△	○	△	○	
		コスト計画、スケジュール作成	○	△	○	△	○	
		チーム編成，要員割り当て		○		△		○
	モニタリング	規模変動把握	○	△	○	○	○	
		進捗，コスト変動把握	○	△	○	○	○	
		ヒアリングや重点的監視		○				
	分析	原因分析		○	○	△		○
	判断	判断		○				
	コントロール	仕様（規模）調整		○				
		生産性，工数，要員再見積り	○	△	○	△	○	
		体制・スケジュール変更		○	○			○
ステークホルダ		ステークホルダマネジメント		○	△			○
リスク		リスク特定，対応		○	△			○
コミュニケーション		計画，情報配付，管理		○	△	△		○

図 6-10　AI 適用の可能性によるアクティビティの分類例

図 6-10 は、図 6-8 に対して、図 6-9 の分類の考え方を適用することによって、「システム化が可能」、「AI で代替できる可能性がある」または「AI で補完できる可能性がある」の観点で分類し、アクティビティごとに該当する分類に「○」を付したものである。ここでの AI は、次節で示す(1) 推論・探索、(2) 知識表現、(3) 機械学習、(4) ディープラーニングのいずれかの技術の意味で用いている。

図 6-10 より、暗黙知の割合が少なくデータ入手が容易なアクティビティについては、システム化できる可能が高いため、「システム化」の欄に「○」または「△」を付している。大島・丸山（2017）の変動可視化手法のように、全面的なシステム化が可能なモニタリング（規模や進捗・コストの変動管理）には「○」を付し、それ以外については、部分的なシステム化が可能という意味で「△」を付している。例えばソフトウェア開発における規模見積りやコスト計画等及びそれに基づく WBS 作成については、定量データの蓄積や見積りの定式化が不十分な場合には、対象業務の有識者の暗黙知が必要となるなど、完全なシステム化は困難である。このような場合は、システム化はマネジメントの補完に止まるという意味で「△」を付しているものである。ただし、これらの「○」や「△」などの程度は分野によって異なる。例えば、プラントエンジニアリングや建設業など、規模見積りやコスト計画の間の定量的な因果関係が得られている分野では、すでにシステム化されていることもあり、「○」となる。

次に、暗黙知が「○」となっているアクティビティは、図 6-11 に示すように、形式知化やシステム化が難しいアクティビティであるが、利用データの欄が「○」のアクティビテ

ィは、蓄積データの量が多くなればAIによって補完できる候補となるという意味で「AIで補完」の欄に「○」を付している。ただし、実際にAIの適用が可能か否かは、アクティビティごとに実現性の検証が必要である。

その他のヒアリング、判断、仕様調整等については、「利用データ」の欄が空欄になっており、これは利用データの入手が難しいアクティビティであることを意味している。これらのアクティビティは、暗黙知の占める割合が多くデータの入手も難しいため、AIの適用も現状では難しいとして、「AIで代替」の欄も「AIで補完」の欄も空欄としている。これらの分類案に基づく具体的なAI技術の適用方策について次節に例示し、今後の研究に資することとする。

6.6. 知識構造モデルを活用したプロジェクトマネジメントへのAI技術適用の考察

提案した知識構造モデルを元にして、ソフトウェア開発のプロジェクトマネジメントにおける具体的なAI技術の適用方策や実現方法について考察する。

ここで、AIについては厳密な定義が定着しているわけではないため、本稿で扱うAI技術の範囲を示しておく。

6.6.1. 実現技術視点でのAIの分類

以下は、総務省の公開資料に基づく、実用面から見た主なAIの研究分野である。

(1) 推論・探索

推論は、人間の思考過程を記号で表現し結論を導くものであり、探索は解く必要のある問題をコンピュータに適した形で記述し、探索木などの手法によって解を得る手法である。

(2) 知識表現（エキスパートシステム）

専門分野の知識を取り込んだ上で推論することで、その分野の専門家と同じ解を導くことを目指すものである。

(3) 機械学習

コンピュータがデータから潜在的なルールや知識を学習することで特徴を捉え、定量化する。学習によって識別したパターンを用いて、新たなデータについて予測を行う技術である。

(4) ディープラーニング

ニューラルネットワークを用いた機械学習の技術の一つで、情報抽出を多階層にわたって行うことで高い抽象化を実現する。ディープラーニングは、予測したいものに適した

特徴量を大量のデータから自動的に学習することができる点が特長である。

プロジェクトマネジメントにおけるこれらの技術の適用場面としては、次のような例が上げられる。例えば、(1) の「推論・探索」は、過去のプロジェクトの実績データからの工数、開発期間の予測や、蓄積した知識の検索などに利用できる。(2) の知識表現は、過去の教訓に基づいたリスクや対策の抽出への応用が考えられる。(3) の機械学習や、(4) のディープラーニングは、プロジェクトの異常検知や所用工数の予測などに応用可能と考えられる。そこで、本稿では、プロジェクトマネジメントに応用可能な AI 技術として、上記の(1)から(4)を対象とする。

6.6.2. 利用用途の観点での AI の分類

安宅（2015）は、機械学習を中心とした AI の利用用途は表 6-1 のように識別、予測、実行に分類している。プロジェクトマネジメントへの適用可能な例としては、例えば「識別」に関してはプロジェクトの異常検知や情報検索などの用途が挙げられ、「予測」に関しては工数や開発期間の予測、「実行」については要員配置の最適化などが挙げられる。

表 6-1　機械学習による AI の利用用途

用途	識別	予測	実行
サブ用途	・情報の判別、仕分け、検索（言語、画像） ・音声、画像、動画の認識 ・異常検知、予知	・数値予測 ・ニーズ予測 ・マッチング	・表現生成 ・デザイン ・行動の最適化 ・作業の自動化

（安宅和人「人工知能はビジネスをどう変えるか」、ハーバード・ビジネス・レビュー2015 年 11 月号、P.47、図表 2 より）

6.6.3. システム化や AI 適用の観点での知識分類

前章で示した AI 技術をプロジェクトマネジメントに応用するにあたって、利用可能なデータと、各データの活用によってどのような AI 技術が適用可能かという観点で具体例を挙げて考察する。

6.6.3.1　AI 適用時のインプットデータの検討

図 6-8 の各アクティビティに対して「利用データ」の有無と「データの例」の欄を追加したものを図 6-11 に示す。

「利用データ」の欄には、各アクティビティに AI を適用するために必要となるデータの入手の容易性を示している。「○」はデータの入手が容易であり、「△」は部分的なデータ

の入手は可能なことを示す。「データの例」の欄には、利用可能なデータの例を挙げており、その詳細を以下に記述する。この際、図 6-3 の知識構造モデルを参照することで、プロジェクトマネジメントの各プロセスでどのような情報を利用できるかを検討しやすくなる。以下に具体例を示す。

(1) 知識構造モデルで示す「管理要素」に関するデータ

以下のような管理要素の指標やそのインプットとなるデータは、システム化や AI 技術の中の推論・探索や機械学習に利用可能である。

(a) 各種管理要素：規模、生産性、要員数、工数、進捗

(b) 品質記録：レビュー指摘件数、テスト件数、障害発生率、ＱＡ数等

(c) プロジェクト単位の管理データ：契約金額、プロジェクト損益、契約回数、契約変更回数、勤務時間[1]

サブジェクト	プロセスグループ	アクティビティ	形式知	暗黙知	利用データ	データの例
統合	計画	規模見積り	○	△	○	規模、画面
スコープ		WBS作成	○	△	○	WBS、生産性
		コスト計画、スケジュール作成	○	△	○	生産性、期間
資源		チーム編成，要員割り当て		○		
タイム	モニタリング	規模変動把握	○	△	○	規模
		進捗，コスト変動把握	○	△	○	進捗、工数
コスト		ヒアリングや重点的監視		○		
品質	分析	原因分析		○	○	各種定量データ
	判断	判断		○		
調達	コントロール	仕様（規模）調整		○		
		生産性，工数，要員再見積り	○	△	○	生産性、工数
		体制・スケジュール変更		○	○	要員数、期間
ステークホルダ		ステークホルダマネジメント		○	△	ステークホルダ特性
リスク		リスク特定，対応		○	△	プロジェクト特性
コミュニケーション		計画，情報配付，管理		○	△	要員特性

形式知化

暗黙知

形式知化

暗黙知

図 6-11　利用可能なデータ

(2) 非構造化データ

ソフトウェア開発プロジェクトの成果物やマネジメント用のドキュメントなどの非構造化データとして、下記の物が挙げられる。これらは、キーワードの集計などに利用可能である。

(a) 各種文書：設計書（ページ数や文字数）、ソースコード（行数、メソッド数）、議事録、課題一覧、変更管理台帳、ＱＡ管理台帳、レビュー記録、テスト仕様書、障害票、障

害一覧（原因区分別件数）

(b) 構成管理情報：資産の格納頻度、資産の変更回数

(c) コミュニケーション関係データ：メール発信数、質問件数、議事録、メール内容

(3) プロジェクト、組織、ドメイン、要員の特性

以下の情報は、暗黙知が占める部分が多い組織、文化やステークホルダ、コミュニケーション等のプロセスに関するものであるが、データとして入手することが難しいものが多い。

(a) 組織やステークホルダの特性情報（組織構造、文化）

(b) 技術特性：開発方式、実装方式、利用技術

(c) 業務特性

(d) 要員特性：業務知識の保有、スキル、パーソナルタイプ等

6.6.3.2 AI 技術の種類別適用方法の検討

前項のインプット情報を元にして、AI をどのように適用することができるかについて、AI 技術の種類別に具体例を列挙する。

(1) 知識化や推論・探索への適用

プロジェクトマネジメントにおける知識の蓄積・活用や事例の検索等において、AI による代替や補完が可能な例として、以下の例が挙げられる。

(a) プロジェクト計画作成やコントロールに関する過去事例の検索や成果物の再利用

(b) 変動の原因分析や判断の支援

(c) リスクの抽出支援や、各リスクへの対応策の提示

(2) 機械学習の適用例

プロジェクトマネジメントに対して機械学習が適用可能な例としては、次のような活用方法が考えられる。

(a) リスクの特定や重み付け

(b) プロジェクトのシミュレーションによる最適な要員投入方法等の検討

(3) ディープラーニングの適用例

ディープラーニングの適用が可能な例としては次のような活用方法が考えられる。

(a) 問題悪化の予兆検知

(b) 異常の発見

6.6.4. プロジェクトマネジメントへの AI 適用の可能性の考察における知識分類モデルの応用

図 6-10 で示したシステム化が可能なアクティビティのうち、変動に関するマネジメント

の例が大島・丸山（2016b）よって提示されている。以下では、図 6-10 で AI による代替や補完が可能と想定した領域について、具体的な実現方策の検討結果や文献に基づいた研究結果を示すことによって、提案した知識分類モデルの活用方法を示す。

6.6.4.1　AI による代替としての適用方策の考察

始めに AI で代替できる可能性のあるアクティビティについて、どのように AI を適用できるかと、そのために必要となるデータについて考察する。

表 6-2　AI で代替できる可能性のあるアクティビティにおける AI 適用方法の例

No.	プロセスグループ	アクティビティ	AI で代替	インプットデータ 教師データ	AI 適用方法	例
1	モニタリング	規模見積り	○	各種規模計測法のインプットとなるデータや画面レイアウト等の情報と見積り結果データ	画面レイウトの情報からの規模見積りの推定や規模変動の検出（ディープラーニングの適用による）	例 1-1
2		コスト計画	○	開発手法別の標準 WBS と、WBS 別の生産性（単位規模当たりの工数）	WBS の策定と WBS 別の基準工数の予測 （機械学習の適用）	例 1-2
3		スケジュール作成	○	同上	No.1、No.2 の AI 適用方法と同じ	（例 1-2）
4		規模変動把握	○	例 1 に同じ	No.1 の AI 適用方法と同じ	（例 1-1）
5		進捗、コスト変動把握	○	進捗、コストの変動データ	進捗やコストの異常検知と推定 （機械学習の適用）	（例 1-2）
6	コントロール	生産性、工数、要員再見積り	○	規模や生産性の変動データと期間及び要員数	要員数の再見積りや要員投入の最適化（シミュレーションと機械学習の適用）	例 1-3

表 6-2 は、図 6-11 の中から、「AI で代替」の列に○を記したアクティビティを抜粋したものである。各アクティビティについて、統計処理や機械学習のために利用可能なインプットデータや教師データの例と、AI 適用方法の具体例を記載している。

以下では、表 6-2 に提示した AI 適用方法の具体的な方策の例を元に AI の適用可能性を評価する。表 6-2 の右列に記載している各例の番号は、以下の例の番号に対応している。

(1)　例 1-1：規模見積りと規模変動の可視化への AI 適用

大島・丸山（2017）によって、規模変動の可視化手法のプロジェクトマネジメントへの活用方法を提案されている。この手法は、規模に関するデータの集計から可視化までの一連のプロセスを標準化したものであるが、機能別の難易度を加味した規模見積りの補正や、規模分布の変動による生産性への影響の評価等は、経験者の知識に依存している。これに対して、表 6-2 の No.1 に示すように、ディープラーニング等の技術を適用することによって画面レイアウト情報や設計情報から機能規模を推定することができれば、規模の計測や変動把握の精度を高め、規模見積りや規模の変動把握を AI で代替できる可能性がある。ただし、この実現のためには画面ごとのレイアトと規模が対となったデータを大量に蓄積することが必要である。

(2)　例 1-2：コスト見積り、コスト計画への AI 適用

大島・丸山（2017）によって、コストや進捗などのプロジェクトの変動を可視化する手法が提案されている。これは標準 WBS と過去の WBS 別の生産性を元に WBS 別の基準工数を算出し、この基準工数をベースラインとして工数の変動や進捗の変動をモニタリングするプロセスをシステム化したものである。ただし、過去のプロジェクトを元にした生産性の設定や WBS のカスタマイズは組織内で有識者が手作業で実施しており、経験が必要な作業である。これに対して、表 6-2 の No.2 に示すように、開発手法に応じた WBS 別の工数や規模の情報を蓄積することで、WBS 別の工数の見積り精度を向上させることができる。また、表 6-2 の No.5 のように、進捗やコストの変動データを分析することによって異常な変動パターン等を検知することができれば、より早期の対策が可能となり、最終的にはモニタリングプロセスの多くを AI で代替できる可能性がある。

(3)　例 1-3：工数や要員数の再見積りへの AI 適用

大島・丸山（2016b）によって、規模の増加や期間の圧縮等によってスケジュールの変更が必要となった場合の要員数を見積もるための再見積り手法が提案されている。この手法は、開発手法が類似した過去のプロジェクトの実績データを元に人が分析した結果を用いている。また、要員の投入時期などは、プロジェクトマネジャーの経験で判断している。これに対して、岡田（2017）が提案しているシミュレーションと、過去のプロジェクトの規模、工数を用いた機械学習を組み合わせることで、最適な要員数や要員の投入時期などの判断に活用でき、投入工数の見直しや要員数の見積りの部分を AI で代替できる可能性がある。このためには、プロジェクトの規模、工数、期間、要員数とその変動状況のデータを、組織的に継続して蓄積してゆくことが重要となる。

6.6.4.2　AI による補完としての適用方策の考察

次に、プロジェクトマネジャーの経験に基づく暗黙知に依存していたようなマネジメントのアクティビティのうち、AI による補完としての適用が可能な例について検討する。

表 6-3 に、AI で補完できる可能性のあるアクティビティにおける AI の適用方法の例を示す。表 6-3 は、図 6-10 の中から、「AI で補完」の列に○を記したアクティビティを抜粋したものである。各アクティビティについて、利用可能なインプットデータや教師データの例と、AI 適用方法の具体例を記載している。以下に、具体的な適用方策の例を元に各 AI の適用可能性を評価する。表の右列に記載している各例の番号は、以下の番号に対応している。

表 6-3　AI で補完可能なアクティビティにおける適用方法の例

No.	プロセスグループ	アクティビティ	AI で補完	インプットデータ 教師データ	AI 適用方法	例
1	計画	チーム編成、要員割当て、WBS 別計画作成	○	規模分布、要員スキル、期間の変動データ	要員の最適な配置や計画変更時のオーバーヘッドの算出	例 2-1
2	モニタリング	ヒアリングや重点監視	○	・設計書、ソースコードなどの開発資産の量の変動データや、メール発信量等 ・プロジェクトの規模や契約、原価等の時系列の変動データ	異常の検出 問題発生の予兆検知	例 2-2
3	分析	原因分析	○	各管理要素の相互関係と各データの変動状況	変動の原因分析支援	（例 2-2）
4	コントロール	体制・スケジュール変更	○	規模、生産性、要員数、期間、スキル	要員再見積り、投入最適化	（例 2-1）
5	ステークホルダ	ステークホルダマネジメント	○	ステークホルダの特性と発生事象	ステークホルダの特性に応じたリスク抽出支援	例 2-3
6	リスク	リスク特定、対応	○	・プロジェクト特性と重大リスクの関係情報と事前対策の例	(a) リスク抽出と対策の準備の支援 (b) 第三者によるリスクの予兆検知	(a) 例 2-3 (b) （例 2-2）
7	コミュニケーション	計画、情報配付、管理	○	コミュニケーション情報（メール、ＱＡ）	異常の検出	（例 2-2）

(1)　例 2-1：工数や要員数の再見積りへの AI 適用

大島・丸山（2017）によって、ソフトウェアの機能規模、生産性、要員スキル等を元に、機能別、WBS 単位の工数計画を一括作成する手法が提案されている。ただし、規模の大小や開発期間に応じた最適な要員配置などは、プロジェクトマネジャーの経験に依存してい

る。表 6-3 の No.1 に示すような規模分布、要員スキル、開発期間等のデータと過去の生産性データ等を元に、最適組合せ問題の解法などを適用することで、最適な要員とタスクの組合せの候補を提示することが可能となると期待される。実際には要員の割り当てには、育成などの観点も含めた判断が必要となるため、AI によって出力される組合せパターンの案は、プロジェクトマネジャーの判断を補完する使い方になると考えられる。

(2)　例 2-2：異常検知への AI 適用

大島・丸山（2017）によって、詳細 WBS 単位の変動を可視化することで、変動の原因特定を支援する手法が提案されている。この手法におけるモニタリングは、基準工数をベースラインとした変動データのみを元にしている。これに加え、表 6-3 の No.2 に示す、設計書やソースコードなどの成果物の量の変動を学習させることにより、プロジェクトの異常の検知ができる可能性がある。また、アジャイル開発手法における継続的インテグレーションでは、開発資産の構成管理情報や変更管理情報を開発支援ツールで蓄積する方法が、標準的な方法として採用されていることが多いため、これらの蓄積データを利用することによる異常検知も期待される。大峡ほか（2017）によって、プロジェクトの会計データ、調達データ、要員数、勤務時間などのデータをディープラーニングや決定木によって学習及び分析することにより、不採算やリスク発生の予兆を検知する方法が発表されている。

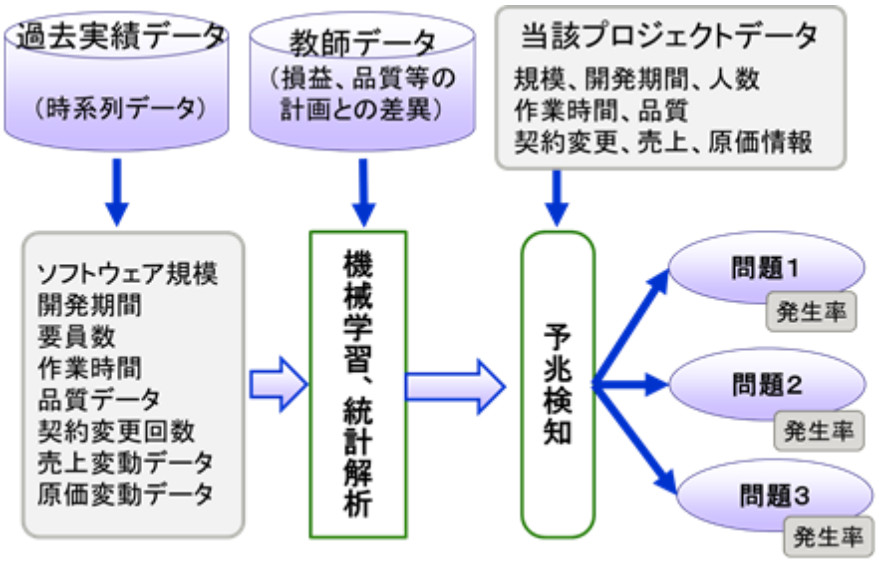


図 6-12　プロジェクトの状況悪化の予兆検知

図 6-12 に、このようなプロジェクトのデータから異常や問題発生の予兆を検知するための実現方式の一案を示す。また、表 6-3 の No.3 に示すように、管理要素の間の関係や成果物の量と工数の相関関係等の分析によって原因分析を支援することも可能と考える。

図 6-12 は、過去のプロジェクトの規模、開発期間、要員数等のデータを入力データとし、損益や品質を元にしたプロジェクトの成功か失敗かの判断を教師データとして学習させることにより、プロエクトの失敗の予兆を検知するための方式である。

(3)　例 2-3：リスクの特定や事前対策への AI 適用

また、表 6-3 の No.5 や No.6 に示すように、過去のプロジェクトの特性を表す情報と発生リスクの情報を蓄積することで、発生する可能性の高い重大リスクを抽出することが可能となる。

富士通（2017）による「KIWare」の中の「プロジェクトリスク予兆検知ツール」において、このような活用方法が発表されている 。この活用方法の実現方式の案を**図 6-13** に示す。

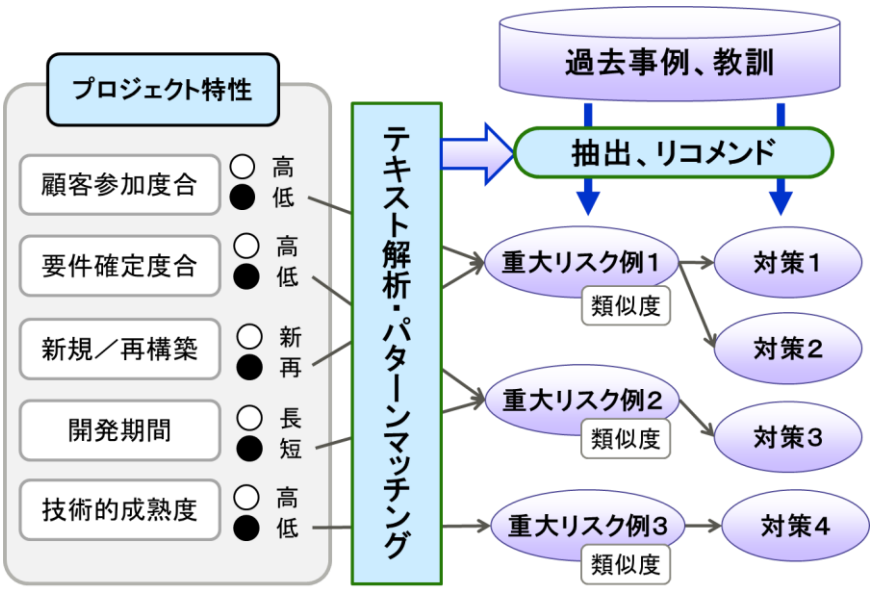


図 6-13　リスクの抽出と対策の助言

過去のプロジェクトから得られた事例や教訓について、各種のプロジェクト特性との関連性に重みをつけてあらかじめ分類しておく。対象プロジェクトの特性を入力すると、テキスト解析やパターンマッチングによって、相関の高いリスクを過去事例から抽出する。さらに、各リスクに対する事前の対策が提示される。これによって、経験の少ないプロジェクトマネジャーであっても、発生する可能性の高いリスクを洗い出し、事前に対策を準備するという活用が可能である。なお、ここで適用している AI 技術は、キーワードの出現頻度などの統計分析の範疇に属するものである。

6.7. 小括

本章では、プロジェクトマネジメントの知識構造モデルを提案した。このモデルの応用例として、知識領域別に、形式知や暗黙知に分類し、プロジェクトマネジメントプロセスにおける具体的なアクティビティを取り上げ、システム化や AI 技術の適用によって代替または補完できる可能性の有無の観点で考察を行った。

なお、事例による考察の過程でも明らかになったように、AI 技術の適用のためには、学習や分析のためのデータの蓄積が重要である。この際、データの種類や粒度が標準化されていなければ効果的な分析や学習が行えないため、プロジェクトマネジメントのプロセスやデータ収集方法の標準化が必要である。

本稿で提案した知識構造モデルは、ソフトウェア開発プロジェクトにおけるウォーターフォール型の開発プロジェクトを前提として整理したものである。ただし、他の分野やアジャイル開発などの場合にも 5 つの知識レイヤーや、知識の種類自体は同様の考え方が適用でき、分野や開発モデルの違いによって部分的に修正することで応用可能であると考える。次章で、より詳細な考察を行う。

7. 考察

本研究で提案した変動マネジメント手法と知識構造モデルについて、先行研究との関係や、応用範囲、課題等の観点から考察する。

7.1. 本研究の新規性と先行研究との関係

第 3 章で述べたように、ソフトウェア開発プロジェクトの定量的管理に関する先行研究においては、ソフウェアの規模や複雑性の定量化手法、各種メトリクスの計測手法、工数見積り手法、及び進捗やコストの定量的把握手法等について、主として精度の向上を目的とした研究が行われてきた。

しかし、実際のソフトウェア開発プロジェクトでは、要件の不確定性や人のスキル差、技術の多様化などの各種要因による影響が大きいことから、メトリクスや見積りの精度を向上させようとしても、進捗や工数の変動が大きくなることが多い。また、このような変動に対してプロジェクトをコントロールするための知識は、プロジェクトマネジャーの経験に基づく暗黙知となっていることが多く、知識の継承が難しい。

本研究は、メトリクスや見積りの精度向上自体ではなく、メトリクスを活用して変動を早期に把握し、変動の原因に応じた適切なコントロールを行うため知識継承に焦点を当てた研究である。プロジェクマネジメントの知識継承を目的とし、定量的管理のシステム化、自動化と知識の形式知化の組合せによって過去の知識を活用する変動マネジメント手法と、知識の形式知化の枠組みとしての知識構造モデルを提案した点に新規性がある。

変動マネジメント手法としては、進捗・コスト等の変動可視化の自動化と、変動の原因に応じたコントロールや再見積りの方法のパターン化を組み合わせ、過去の知識を形式知化して活用するマネジメント手法を提案した。合わせて、変動マネジメントに必要な具体的な知識に焦点を当て、プロジェクトの監視・制御のプロセスとの関連性の視点での知識の構造や知識要素間の関係を一般化した知識構造モデルを提案し、形式知化すべき知識領域の特定や知識の形式知化の方法を検討するための枠組みとした。

以上のように、本研究の新規性は、変動可視化のシステム化とコントロール方法のパターン化を組み合わせることで、過去の知識を形式知として活用する変動マネジメント手法と、知識の形式知化の枠組みとしての知識構造モデルにより、知識継承を促進する新たな方法を提案した点にある。

以下では、提案した変動マネジメント手法と知識構造モデルのそれぞれについて、先行研究との関係や具体的な特徴及び新規性について示す。

7.1.1. 変動マネジメント手法の新規性について

提案した変動マネジメント手法の新規性と、先行研究との関連性について表 7-1 に示した後、考察を加える。

表 7-1　変動マネジメント手法の新規性

No.	本研究の新規性	関連する先行研究
1	変動の可視化手法 ・計画時の精度向上に加え、変動を早期に把握して計画と実績の差を少なくなるようにコントロールすることに主眼を置き、先行研究で提案されている各種メトリクスの活用方法としての監視と制御の手法を提案した点に新規性がある。 ・変動可視化手法（デイリーEVM 手法）の特徴は、下記の 2 点である。 (1) 詳細な粒度の WBS の単位での日々の進捗と工数の変動可視化を、基準工数の概念と生産性や習熟度等のパラメータによる工数計画作成の一括作成と、実績工数を元にした進捗率の自動集計等により、運用負荷を増やすことなく実現した。 (2) ソフトウェア規模を定期的に集計するプロセスを標準化することによって規模の変動を可視化し、進捗や規模と合わせ一体として把握可能とした。	・先行研究においては、マネジメントの正確性を向上させるために必要なメトリクスの精度向上を目的とした研究や、進捗とコストを俯瞰的に把握するための EVM 手法の改善に関する研究が行われている。しかし変動の早期把握等のためにメトリクスを活用するという視点は含まれていない。 ＜先行研究の例＞ ・ Anbari (2003) 、 Fleming and Koppelman (2003)、Cooper et al. (1993)、Pajares (2011)、Elshaer (2013) による EVM の拡張や精度に関する研究 ・Demirors (2013) によるソフトウェア開発プロジェクトへの EVM 手法の研究 ・Putnam and Myers (1995) による見積りの研究 ・Garmus and Herron (1996) 、Garmus and Herron (2000)、Jones (2013) による FP 法の研究 ・McCabe (1976) による複雑度等の研究 ・Kirmani (2017) による Web システムにおける早い段階での工数見積りを精度良く行うための研究
2	変動のコントロールに関する知識の形式知化 ・メトリクスを変動の原因分析やコントロールに活用すること、及び変動の原因に応じた適切な対策のための知	・定量的マネジメントに関する先行研究は、計画の精度向上や実績の評価を目的としているものが多い。メトリクスをコントロールに

	識を形式知化し、マネジメント手法に組み込んだ点に新規性がある。 具体的には以下の 2 点である。 (1) 経験に基づく暗黙知となっているコントロール方法に関する知識を、パターン化、ルール化することにより、形式知化した。 (2) プロジェクトマネジャーの経験や暗黙知に依存していることが多い再見積りに関する知識として、見積り時に考慮している作業の習熟度や要員追加によるオーバーヘッドなどの典型的な工数増加要因をパターン化し、各要因による工数への影響度合いをあらかじめ見積り式に組み入れることで、見積り手法として形式知化した。	活用する視点や、プロジェクトマネジメントの熟練者の知識の形式知化の視点は含まれていない。 ＜先行研究の例＞ ・知識の内面化のための Hawk (1999) の学習に関する研究や、内田（2016）による開発プロジェクトマネジメントの知識を教訓として継承する方法の研究はあるが、マネジメントに直接利用可能な知識の形式知化に焦点を当てたものはない。 ・工数見積に関する先行研究は、Putnam and Myers (1995) による見積りの研究等の、工数見積り精度の向上を目的としたものである。 ・Kirmani (2017) は、段階での工数見積りの要件として、3 つの要件（概算とを実績の差異が少ない、監視やプロセスの改善、個人レベルでの見積り根拠の明確化）を挙げている。本手法での工数見積りはこの考え方に沿っており、さらに、過去の経験を形式知として取り入れたものである。 ・再見積りに関する研究としては、MacDonell (2003) による、前工程の工数を入力情報とした回帰分析によって次工程の工数を見積もる手法、Ferrucci et al. (2010) による前工程の工数実績と合わせて Function Point を組み合わせた見積り精度向上、Azzeh et al. (2010) による、相関ルールやファジー集合の概念を利用した工数見積り精度の向上、Emran et al. (2010) によるソフトウェアのリリース計画の再作成において、機能性と品質向上のトレードオフを行うためのシミュレーション、前工程の情報から次工程以降の工程別の工数を見積もる際の精度を向上させることを目的としたものがあるが、プロジェクトマネジャーの経験に基

		づく規模の変動や計画の変更自体が、期間短縮や要員追加、作業分割などの工数増加に及ぼす影響を考慮した研究は行われていない。

表 7-1 のように、先行研究では、ソフトウェア開発プロジェクトの定量的なマネジメントの正確性を向上させるためには、精度の高い見積りや計測手法の改善が必要という視点で研究が行われてきたと言える。しかし、精度の向上を求めるとデータ収集や検討作業等の負荷が高くなり、マネジメント実行上の阻害要因となる。また、実際のプロジェクトにおいては、人のスキルや要件の確定度合いなどによるばらつきが大きいことから、意図した精度が得られず、結果的に計画に対する変動が大きくなるため、変動への対処の遅れによるコスト増加やスケジュールの遅延が発生するリスクがある。

そこで、本研究では、計画段階における精度の向上に加え、プロジェクトの変動を早期に把握し、変動が拡大する前に変動の原因に応じたコントロールを行うことでプロジェクトの成功率を向上させることに主眼を置き、変動の監視と制御の手法を提案した。以下に提案した変動マネジメント手法を構成する 3 つの手法別の新規性について述べる。

7.1.1.1. 変動の早期把握に視点を置いた変動可視化手法（デイリーEVM 手法）

プロジェクトの定量的管理に関する先行研究としては、表に示すように、プロジェクトの規模や品質などの状態を定量化するための各種メトリクスの精度向上を目的としたものや、プロジェクト全体の進捗とコストの俯瞰的な管理手法としての EVM 手法等の研究が行われてきた。

本研究では、精度の向上に加え、変動を把握するためにメトリクスを活用し、早期に生産性や各種のパラメータの補正を行うことによって、計画の精度を向上させ、正確かつ詳細な情報に基づいて、計画と実績の差異を早期に抑制することを可能とした点に新規性がある。

具体的には、提案したデイリーEVM 手法は、先行研究に対して下記の(1)と(2)の 2 点において新規性がある。

(1) 詳細 WBS 別の基準工数と進捗率自動計上による変動の可視化と原因分析への活用

従来の EVM 手法は、週次や月次のサイクルでプロジェクト全体としての俯瞰的な進捗やコストを把握することを主目的としたものであり、一つの WBS の期間が数日にわたる粒度で計画が策定されていた。詳細な WBS や機能の単位での進捗やコストの変動を、日次等の短いサイクルで管理しようとすると運用負荷が高いために実現できなかった。

本研究では、詳細 WBS 別の基準工数と、開発作業と一体となった実績工数の入力値だけから進捗率を自動集計可能な進捗計上ルールを考案することによって進捗率の計上や異常値の可視化を自動化し、データの集計や可視化のための運用負荷の抑制と精度の向

上を両立させた点が特徴となっている。具体的には、一つの WBS を数時間程度の粒度とした WBS 別の単位工数の設定、人別のスキルや習熟度の考慮、詳細 WBS と合わせた進捗ルールの設定などの本手法独自の仕組みによって、進捗率の自動計算を可能とした。工数計画についても、規模、生産性、習熟度などのパラメータから一括作成可能としたため、実績データを元に計画を補正することが容易であり、逐次、計画の精度を高めることができることを検証した。

先行研究の例としては以下の研究があるが、いずれも、EVM 手法の改善や工数見積り精度の向上のための研究であり、日々の変動可視化や変動の原因分析への活用の観点は含まれていない。

① Anbari (2003) 、Fleming and Koppelman (2003)、Cooper et al. (1993)、Pajares (2011)、Elshaer (2013) による EVM のスケジュール管理への拡張や精度の向上に関する研究

② Demirors (2013) によるソフトウェア開発プロジェクトへの EVM 手法適用の研究

③ Kirmani (2017) による Web システムにおける早い段階での工数見積りを精度良く行うための研究

(2) ソフトウェア規模の変動可視化

ソフトウェア規模に関する先行研究は、新たな規模のメトリクスを提案するものや、規模の見積り精度向上に関するものであった。本研究では、工数や進捗などのプロジェクト管理の基盤を成す重要なソフトウェア規模を、変動のマネジメントに活用するために、計測担当者や実装技術に依存しない安定性や、ライフサイクルの初期での計測可能性、計測の容易性、理解容易性等を重視して計測手法を選定し、規模の変動や分布を定期的に集計、可視化することによって、進捗・コストと一体として変動を把握可能とした。

規模に関する先行研究としては、以下の研究があるが、新たなメトリクスの提案や、規模の精度向上に関するものが主体であり、規模の変動把握方法の視点は含まれていない。

① Putnam and Myers (1995) による各種の規模見積りのメトリクスや手法の研究

② Garmus and Herron (1996) 、Garmus and Herron (2000)、Jones (2013) による FP 法の研究

③ McCabe (1976) による複雑度等の研究

7.1.1.2. 変動のコントロール手法に関する知識の形式知化

定量的マネジメントに関する先行研究には、前述のようにメトリクスや見積りの精度向上を中心とした研究が多く行われている。しかし、各メトリクスをコントロールに活用する方法や、プロジェクトマネジメントの熟練者の知識をどのように形式知化するかについてはそれらの先行には含まれていない。

本研究の変動に対するコントロール手法は、過去のプロジェクトマネジメントの知識継承を目的とし、過去の実績と人間のノウハウを、変動可視化手法の活用と合わせてパターン化、ルール化することによって、形式知化した点に新規性がある。これによって早期の問題発見や変動の原因に応じた対処を行うことを可能とした。

(1) 知識継承及び知識の形式知化について

知識継承に関連した先行研究としては、下記のような先行研究があるが、マネジメントに直接利用可能な知識の形式知化に焦点を当てたものはない。

① 知識の内面化のための Hawk (1999) などの学習に関する研究

② 内田（2016）.による開発プロジェクトマネジメントの知識を教訓として継承する方法の研究

(2) 工数再見積方式について

工数見積りや再見積りに関する先行研究として、下記のような多くの研究があるが、前工程の情報から次工程以降の工程別の工数を見積もることによって見積り精度を向上させることを目的としたものであり、プロジェクトマネジャーの経験に基づく規模の変動や計画の変更が、期間短縮や要員追加、作業分割などの工数増加に及ぼす影響をパターン化することによる工数の再見積り等の研究は行われていない。

① Putnam and Myers (1995)　による見積りの研究等の、工数見積り精度の向上を目的としたもの

② MacDonell (2003) による、前工程の工数を入力情報とした回帰分析によって次工程の工数を見積もる手法

③ Ferrucci et al. (2010) による前工程の工数実績と合わせて Function Point を組み合わせた見積り精度向上

④ Azzeh et al. (2010) による、相関ルールやファジー集合の概念を利用した工数見積り精度の向上

⑤ Emran et al. (2010) によるソフトウェアのリリース計画の再作成において、機能性と品質向上のトレードオフを行うためのシミュレーション

⑥ Kirmani (2017) は、Web システムの工数見積りを精度良く行うための見積り手法の選定に関する研究

以上のように、本研究で提案したプロジェクトの変動に関するマネジメント手法は、規模や進捗、コスト等の変動を可視化する方法のシステム化と、変動に対するコントロール方法に関する知識のパターン化や形式知化によって知識の継承を促進するためのものである。

提案した変動コントロール手法を知識継承の観点で捉えると、変動の可視化や変動に対するコントロールの知識を、プロジェクト計画の変更や仕様調整などのパターンとして形式知化する手法であると言える。変動の原因に応じてパターン化することによって形式知化された知識は、WBS の作成方法、工数計画作成方法、変動の原因分析方法、変動の原因別のコントロール方法及び再見積方式等である。この際、知識を形式知化するだけではプロジェクトにおけるマネジメントの実践での活用に至らない。このため、プロセスの標準化や知識のパターン化をドキュメント化するに留まることなく、自動化やシステム化及び定式化することによって知識を行動に繋げる仕組みを構築した点が特徴であると言える。例えば、進捗遅延に対する対策を行う場合に、プロセスの問題なのか人のスキルの問題なのかによって対策が異なるため、原因分析のための WBS 別の工数実績を毎日把握可能とする仕組みや、コントロール方法の一つとして要員の追加を行うための工数見積りの定式化手法を構築した。

機能や担当者別の詳細な粒度での変動を可視化し原因分析に活用するため、WBS を詳細化し、人別の生産性を考慮した計画を策定する上で、計画の作成に多大な時間を要しては実行することが難しい。そこで、実行上の阻害要因となる手法の習得や運用に関する負荷を軽減するための対策を手法に組み入れた。具体的には、単位工数を元にした工数計画の自動作成及びパラメータ変更による工数計画の一括変更、実績工数入力と連動した進捗率の自動計上、規模計測の自動化、各種の変動可視化ドキュメントの自動出力等の仕組みがシステム化の範囲となっている。これらの特徴により、人別の詳細 WBS 単位での変動を毎日把握可能とすることが可能となり、早期の問題解決やマネジメントの精度向上を図ることができる。このように、阻害要因となっていた運用負荷の増加を軽減し、実効性の高い手法とした。

また、Kirmani (2017) は、早い段階での工数見積りの要件として、3 つの要件（概算とを実績の差異が少ない、監視やプロセスの改善、個人レベルでの見積り根拠の明確化）を挙げている。本手法での工数見積りはこの考え方に沿っており、さらに、上記のように工数計画作成の自動化や進捗率計上の自動化などの仕組みを加え、過去の経験を形式知としてマネジメントのプロセスに組み込んだものである。

このように、自動化やシステム化は、データを活用可能な環境を整え、マネジメント手法を標準化することによって暗黙知となっていた知識をパターン化して形式知とすることを実現する上での前提条件となるものであると言える。

7.1.2. プロジェクトマネジメントの知識構造モデルの新規性について

表 7-2 に、本研究で提案した知識構造モデルの新規性と関連する先行研究について示す。

表 7-2　プロジェクトマネジメントの知識構造モデルの新規性

No.	本研究の新規性	関連する先行研究
1	知識の構造化 ・先行研究で行われている知識継承の方法やプロセスの構造等の観点ではなく、知識継承の方策検討等を行う際の対象範囲や継承方法を検討する上での枠組みとして、開発プロジェクトの変動のマネジメントに必要な具体的な知識内容に焦点を当て、知識要素の位置付けや階層、要素間の関連性等を表す知識構造モデルを提案した点に新規性がある。これによって、知識の形式知化を行う上での具体的な知識の領域を特定するための枠組みとしたものである。具体的な特徴は以下の 2 点である。 ＜知識構造モデルの特徴＞ (1) プロジェクトを制御対象として捉えたときの監視（モニタリング）と制御（コントロール）の視点から、プロジェクトの状態を表す管理要素とそれを介した変動管理、さらにそれぞれの背景知識となる実践知という形の知識レイヤーに分けて分類した。 (2) 各知識レイヤーや知識要素間の因果関係や、監視と制御のプロセスで各知識をどのように組み合わせて活用するかについての関係を示した。 以上により、プロジェクトマネジメントの知識を、どの場面で、いずれの知識を組み合わせて活用するのかなどが明確になり、組織内での知識継承における具体的な知識の形式知を進める際、対象範囲を設定する上での枠組みとして活用することが可能である。	・知識継承に関する先行研究としては、知識の表出化や継承のプロセスや知識継承の方法や継承のプロセスに関するモデル化等が発表されている。しかし、プロジェクトの変動を把握し、コントロールするといった、マネジメントの実践のための具体的な知識の形式知化を目的とした知識のモデル化に関する研究は行われていない。 ＜先行研究の例＞ ・内田（2016）.による開発プロジェクトマネジメントの知識継承フレームワークの研究 ・内平（2010）による研究開発プロジェクトの知識継承フレームワークの研究
2	知識の分類方法 ・PMBOK や ISO21500 等の既存の知識体系が、マネジメントプロセスの種類によって分類されていたのに対して、本研究では、監視と制御の各マネジメントプロセスに関連づけて各知識レイヤーを	・プロジェクトマネジメントの知識体系としては、PMBOK、ISO21500、PRINCE2 等の知識体系があるが、知識をプロセスや

	階層化した上で、形式知化やシステム化が容易な領域と、暗黙知の占める部分が大きく形式知化が難しい領域に分類した点に新規性がある。これによって、知識レイヤーごとに、効果的な知識継承の方法を選択することが可能となった。	スキルの観点で分類し、ツールや手法の位置付けを整理するに留まっている。 ・Horner and Yong (2006) により、PMBOK ガイドは、形式知や宣言的な知識に偏っており、暗黙知や理由（Why）に関する知識にはあまり注意が払われていないことが指摘されている。

上記表のように、知識継承に関する先行研究としては、知識の表出化や継承のためのプロセスや継承の方法の観点でモデル化するための研究や、知識の分類体系としての整理が行われている。本研究で提案した知識構造モデルは、プロジェクトマネジメントの実践のための具体的な知識の形式知化を目的とし研究である点に新規性がある。

以下に、知識の構造化と分類方法の観点での本研究の新規性について述べる。

(1) 監視と制御のプロセスに対応させた具体的知識の構造化

知識継承に関する先行研究としては、知識の表出化や継承のためのプロセスの解明や、知識の共有や継承の方法及び継承プロセスに関するモデル化等が発表されている。しかし、プロジェクトの変動を把握し、コントロールするといった、マネジメントの実践のための具体的な知識の形式知化を目的とした知識のモデル化に関する研究は行われていない。関連する先行研究としては、下記の研究があり、知識の送り手と受け手の間の構造や、知識の表出化や内面化を効果的に行うためのフレームワークや手法が提案されている。

① 内平（2010）による研究開発プロジェクトの知識継承フレームワークの研究

② 内田（2016）.による開発プロジェクトマネジメントの知識継承フレームワークの研究

本研究で提案した知識構造モデルは、上記①で示されている研究開発のプロジェクトマネジャーに必要な知識構造のモデルを参考にして、プロジェクトマネジメントの具体的な知識の要素の位置付け、構造や関係性をモデル化したものである。プロジェクトを制御対象と見なしたとき、制御対象の監視に関するもの、プロジェクトの制御に関するもの、監視や制御に必要な管理要素、及び経験に依存することが多い実践知等の、どの層に各知識が位置づけられるかを明示するため、知識のレイヤーと、知識要素間の関連性及び制御の流れの視点を取り入れた点に新規性がある。これによって、変動の監視、コントロール、原因分析などのいずれの作業を行う時に必要な知識か、またはどのような管理要素（情報）を使ってコントロールを行うのかなどの観点が明確になる。継承すべき対象となる具体的な知識を特定した上で、知識の種類に応じた形式知化の手法を選定するというように、知

識継承のための具体的な知識の形式知化の枠組みとして活用することが可能となる。さらに、形式知化された知識やマネジメント手法を活用する際に、知識要素間の関連性を参照しながら、変動の原因究明や対策検討の際に、過去の知識を活用することが可能となった。

プロジェクトの変動に対する原因究明や対処方法については、プロジェクトマネジャーの暗黙知の領域と見なされていた部分が多く、他のプロジェクトマネジャーに継承することが難しかったが、本研究で提案した知識構造モデルによって、暗黙知の内容を整理し、パターン化などの形式知化を行うことが可能となった。管理要素や知識間の関係を明確にすることによって、データを収集、活用するための作業を自動化やシステム化し、WBS の分類方法や工数計画の作成方法の標準化と組み合わせることによって、従来暗黙知であった知識を形式知として活用することを可能としたものが、提案した変動マネジメント手法である。

(2) 形式知化、システム化、暗黙知の観点での知識分類

プロジェクトマネジメントの知識体系としては、PMBOK、ISO21500、PRINCE2 等の知識体系があるが、知識をプロセスやスキルの観点で分類し、ツールや手法の位置付けを整理するに留まっている。Horner and Yong (2006) により、PMBOK ガイドは、形式知や宣言的な知識に偏っており、暗黙知や理由（Why）に関する知識にはあまり注意が払われていないことが指摘されている。

本研究では、プロジェクトマネジャーの知識を、知識領域の特性に応じて形式知化またはシステム化が可能な領域と、それ以外の、暗黙知の占める割合が多い知識領域等に分類した点に新規性がある。提案した(1)の知識構造モデルと対応させることによって、形式知化やシステム化が容易なレイヤーと暗黙知の占める割合が多いレイヤーに分けることが可能となった。これによって、組織内での知識の形式知化による蓄積や、システム化によるマネジメントの補完方法の検討に活用可できる。知識分類の活用方法の一例として、AI技術によってプロジェクトマネジメントを補完する方法の検討例を通じて、提案した知識モデルと知識分類の有効性を確認した。

7.2. 本研究の前提条件や応用範囲について

本研究で提案した変動マネジメント手法や、知識構造モデルの適用対象のプロジェクトは、企業内や企業間の取引などに関する情報システムの開発プロジェクトを前提としている。また、開発要員は組織内や協力関係にある企業に所属する同一の要員が参画するプロジェクトを一義的に想定している。このような条件の組織や環境におけるプロジェクトの場合は、WBS の粒度や、生産性、習熟度、スキルなどの各種の定数を決める上での過去のデータの蓄積があるため、これらのデータを再利用することで精度の高い計画作成や実績

データとの比較を行いやすい。この前提条件以外のプロジェクトに適用する場合には、WBS の粒度や各定数をプロジェクトに合わせて見直す必要がある。

この場合でも、本手法の特徴である次の考え方は応用可能と考える。

(1) 標準 WBS の詳細化、標準化と合わせた、各種の管理要素としてのパラメータによる工数算出と工数計画の一括作成、及び進捗率の自動計算

これによって、工程の初期の実績データによってパラメータを見直し、プロジェクトの期間中に計画の精度を向上させることができる。先行研究の多くは、初期の見積り精度を高めるための研究であったが、情報システムの複雑性が増加し、要件が変動しやすい近年の IT 活用システムにおいては、変動を早期に定量的に把握し、計画を早期に修正することが必要となり、このための対策として本手法を応用可能と考える。

(2) 短いサイクルでの詳細な粒度での計画作成と変動把握

短いサイクルかつ詳細な粒度で計画を作成し、変動を把握することが、変化の多いプロジェクトのマネジメントの成功の条件である。

(3) 変動の原因分析やコントロールの知識のパターン化

変動に対するコントロールに関する知識については、プロジェクトマネジャーの経験に基づく暗黙知となっているものが多いが、本手法で提案したパターン化による暗黙知の形式知化は、様々なプロジェクトで応用可能である。この際に、WBS を標準化し、工数計画作成や進捗率の算出などを、変動の要因と関連した管理要素（パラメータ）との関係を明らかにすることによって自動集計できるようにする方法を採用することで、原因分析の仕方と合わせた工数再見積りなどのコントロールのパターン化が可能となる。

(4) 制御対象としてのプロジェクトの視点でのマネジメント知識の構造化

提案した知識構造モデルは、ソフトウェア開発プロジェクトを前提としたものであるが、プロジェクトを制御対象と見なして状態を監視し制御を行うという観点に基づいて知識を構造化し、監視・制御のプロセスと管理要素や知識の間の関係性を明らかにするという、知識の構造化の考え方は、他の分野のプロジェクトマネジメントにも応用可能であると考える。

本研究で提示したこれらの方法やアプローチ方法を、他の分野に拡張するための研究が期待される。

8. 結論

本研究では、ソフトウェア開発プロジェクトにおける規模や進捗、コストの日々の変動を、機能別や人別の詳細な WBS の粒度で運用負荷を増やすことなく可視化し、制御するための一連の手法と、知識の蓄積、継承、活用を目的とした知識構造モデルを提案した。以下に、リサーチクエスチョンへの回答と、研究の理論的、実務的な含意及び今後への展望を述べる。

8.1. リサーチクエスチョンへの回答

第 1 章 1.1 節で示したリサーチクエスチョンへの回答を以下に示す。

SRQ1：プロジェクトマネジャーは、各種の変動をいかなる方法によって負荷低減と精度向上を両立しながら把握することが可能となるのか。

ソフトウェア開発における規模や進捗、コストの変動をリアルタイムに精度良く把握することが困難であるという課題に対して、進捗・コスト（工数）の変動を短いサイクルで定量的に把握するための基準となる詳細な WBS 別の工数計画と、規模を定期的に計測するための仕組みを事前に準備しておくことにより、プロジェクトマネジャーは、運用負荷低減と計画と実績に関するデータの精度向上を両立しながら、日々の変動を把握することが可能となる。図 8-1 に、提案した変動マネジメント手法の全体像を再掲する。

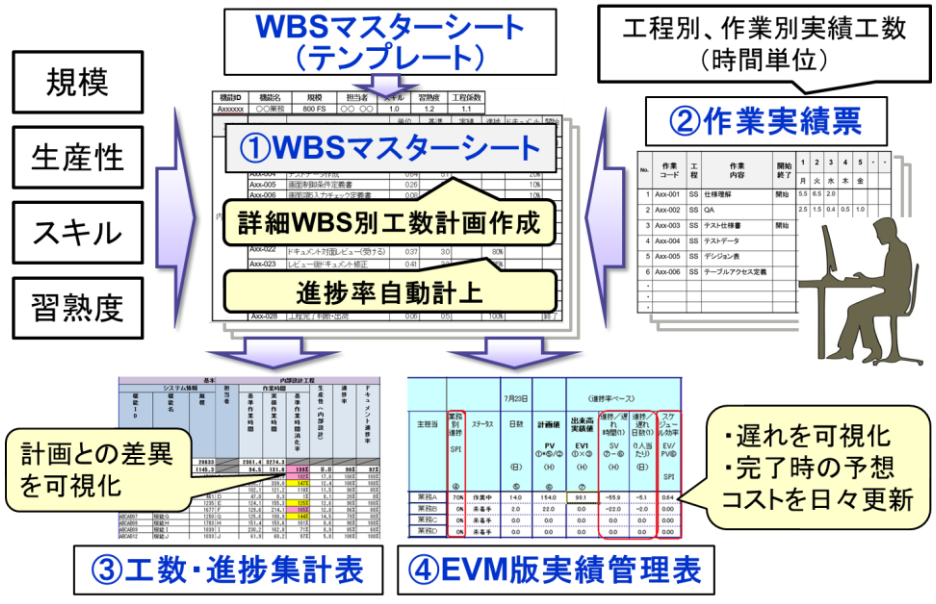


図 8-1　変動可視化手法（デイリーEVM 手法）の全体像（再掲）

本手法は、機能別や人別の詳細な WBS の粒度で進捗、コスト（工数)、規模の変動を定量的に可視化するための手法である。一つの WBS の粒度を時間単位で詳細化し、日々の変動を可視化するためには、運用負荷の増加が阻害要因となるが、以下の 3 点の特徴によって、従来と同等の負荷で実行可能である。

(1) 詳細 WBS と基準工数による工数計画の一括作成

一つの WBS の作業工数が数時間程度の詳細 WBS と、単位規模当たりの WBS 別単位工数を対にした WBS マスターシートのプロジェクト用テンプレートを元に、全 WBS の「基準工数」の計画値を、規模と生産性などのパラメータ入力によって算出し、機能別、人別の工数計画を一括で作成可能にした。この基準工数が変動把握のベースラインとなる。

(2) 進捗計上ルールと実績工数データからの進捗率の自動計上と変動の可視化

各担当者が実績工数を入力するだけで進捗率が毎日自動計上されるよう、作業や工程の特性を考慮して進捗計上ルールを設定しておくことによって、主として実績工数の入力情報から進捗率が自動計上される仕組みを構築した。これにより、出来高（基準工数 × 進捗率）と実績工数の情報から、生産性の変動把握や EVM によるコスト予測が可能となる。進捗計上の作業は、従来は、報告に基づいて手作業で行われていた部分であるが、一つの WBS を数時間程度の詳細な粒度とした上で、WBS 単位の基準工数と対にして標準化し、各 WBS に進捗率計上ルールを埋め込んでおくことによって、自動化が実現できたものである。

(3) 規模変動の可視化

工数の見積りやスケジュール作成の基礎となるソフトウェア規模の変動を精度良く把握するため、本研究では、設計情報を元に定期的に規模を計測し、可視化する手法を提案し、この規模の変動情報を、上記(1)の工数計画作成や(2)の進捗計上のパラメータ情報へフィードバックする方法を提案した。これによって、従来の手法では難しかった規模、工数、進捗の変動を一体として可視化することが可能となった。この際、一連のプロセスを標準化及び自動化することによって、可視化のための管理負荷の低減と変動の精度向上を両立した。

上記の特徴により、詳細 WBS 及び計画値の作成、実績工数の時間単位での収集、進捗率及び出来高の自動計上による日々の変動可視化の一連のプロセスを、従来と同等以下の運用負荷で毎日実行することが可能となる。

SRQ2：プロジェクトマネジャーは、変動に対していかなる情報や知識をどのように活用することでプロジェクトを円滑にコントロールすることが可能となるのか。

プロジェクトのコントールについては、知識に占める暗黙知の割合が多いために属人化し知識継承が難しいという課題があった。この課題を解決する方法として、プロジェクトの変動要因に応じた代表的な対応策をあらかじめ想定し、パターン化によって形式知化した上で、各対策に必要な実績データの把握方法と再見積方式を準備しておくことによって、過去の人のノウハウを形式知化した。これにより、プロジェクトマネジャーは、可視化された変動の実績データを分析し、変動に影響を及ぼしている要因が存在する WBS 等を特定し、原因に対する適切な対策を実施することが可能となる。実施した対策に合わせて工数計画を修正することにより、対策実施後の変動を監視し、実施した対策の効果を早期に確認することによって、変動を抑制し、計画との差異が少なくなるよう、プロジェクトを円滑にコントロールすることが可能となる。

図 8-2 に、変動可視化によるコントロールプロセスを示す（再掲）。

提案した手法では、コントロールに必要な工数やスケジュールの再見積り手法を含む対処方法を、変動要因に応じたパターンとして形式知化することによって、変動のケースに応じた適切なコントロールを可能とした。

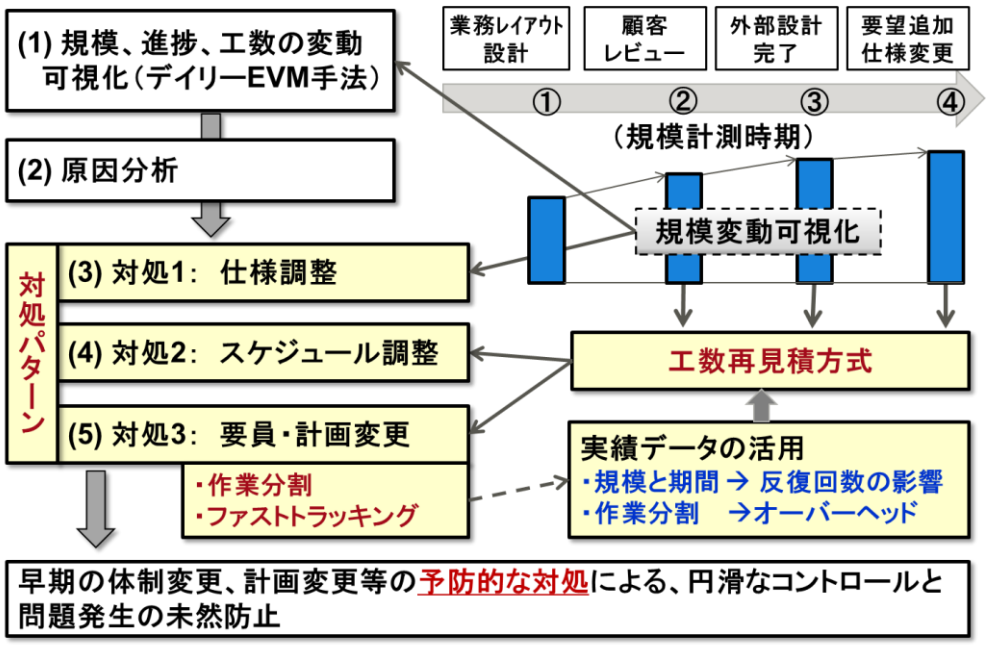


図 8-2 変動可視化によるコントロールプロセス（再掲）

上記のコントロールプロセスのポイントは、プロジェクトの変動要因に応じ、代表的な対応策をあらかじめ想定し、パターン化した上で、各対策に必要な実績データの把握方法と見積方式を準備しておくことである。従来は、計画との差異に対する対処はその都度、経験による見積りや属人的な判断を加味して行われており、個々のプロジェクトマネジャーの暗黙知の領域であったために知識の継承が難しかった。本手法では、変動の原因に応じたコントロール方法をパターン化し、形式知化することによって、経験の少ないプロジェクトマネジャーでもプロジェクトの変動の原因に応じた対策を適時に実施することが可能である。

プロジェクトへの適用によって検証した通り、変動マネジメント手法と図 8-2 のコトロールプロセスの適用によって、精度の高い進捗把握、異常の早期察知、問題発生時の原因究明及び工数予測等を定量データに基づいて行うことができ、問題の未然防止や、円滑なプロジェクトのコントロールが可能となる。

SRQ3：プロジェクトの変動に対処するための知識は、いかなる要素から構成され、各要素はどのような関係性を有するのか。

図 8-3 図及び図 8-4 に、提案した知識構造モデルと知識レイヤー間の動的モデルを示す。

知識構造モデルによって、プロジェクトの変動に対処するための知識は、プロジェクト知識（ドメイン知識）、管理要素、コントロール、変動管理、及び実践知の各知識階層とそれぞれに含まれる知識要素から構成され、各要素の間には、因果関係としての図 8-3 に示す関係や、監視・制御のプロセスとの動的な関係（プロセスと組み合わせて活用する知識の関係）は、図 8-4 に示す関係を有することが明らかになった。

プロジェクトマネジメントに関する知識の蓄積及び継承への活用を目的として、プロジェクトの変動に対する監視と制御の方法や知識を、プロジェクトを制御対象と見なすことによって階層的に整理した知識構造モデルを提案した。プロジェクトの監視と制御のどのマネジメントプロセスにおいて、どの知識を組み合わせて利用するのかという観点での知識階層（レイヤー）間の動的な関係モデルと合わせて利用することで、プロジェクトの変動に対処するための知識の構成と、知識要素間の関係が明確になった。

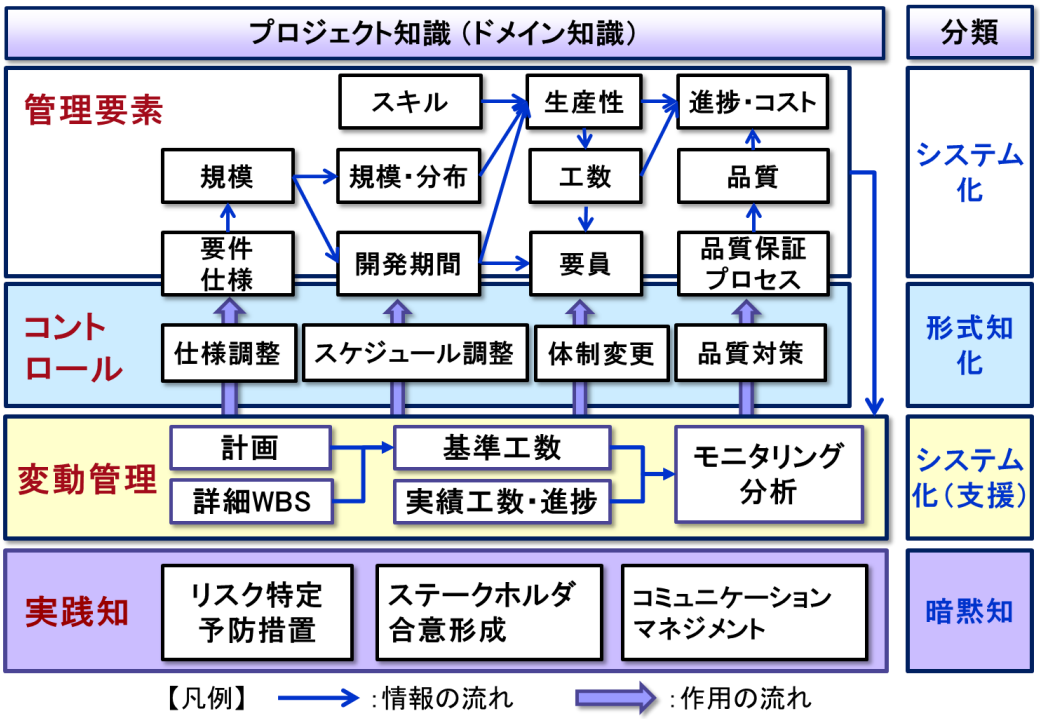


図 8-3　プロジェクトマネジメントの知識構造モデル（再掲）

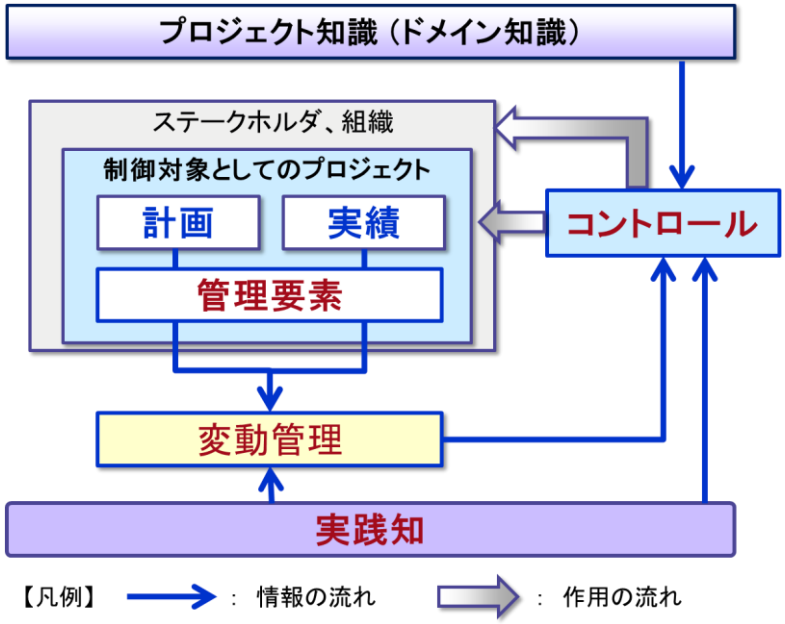


図 8-4　知識レイヤー間の動的関係モデル（再掲）

図 8-3 の知識構造モデル及び図 8-4 の動的関係モデルの特徴は、PMBOK や ISO21500 などのように知識領域やプロセスをサブジェクトグループやプロセスの種類で分類するだけではなく、管理要素や各知識と対処方法の間の動的な関係を構造化し、モデル化した点である。これにより、マネジメントの各プロセスにおいて「いかなる情報を元に、どう対処

すれば良いか」といった観点で知識を活用しやすく、継承すべき知識も明確になり、プロジェクトマネジメントの具体的な知識の可視化による共有や知識継承の促進に活用できる。

また、図 8-3 の右側に示す「形式知化」、「システム化」、「システム化（支援）」及び「暗黙知」という分類によって、各知識のレイヤーごとに、形式知化やシステム化が比較的容易な領域と、形式知化が難しく暗黙知の占める部分が多い領域に分類できる。これによって、各知識レイヤーの特性に応じて、各組織内で効果的な知識継承を行うことが可能となる。このモデルの応用方法の一例として、システム化や AI 技術の適用によって代替または補完できる可能性の有無の観点でプロジェクトマネジメントのアクティビティを分類することにより、利用可能なデータの抽出やプロジェクトマネジメントへの AI 技術の適用方法の検討に利用できる。

【メジャーリサーチクエスチョン（MRQ）】

ソフトウェア開発プロジェクトにおいて、プロジェクトマネジャーは各種変動をどのように定量的に把握し、いかなる知識、経験を活用してプロジェクトをコントロールするのか。

目に見えにくく、変動が発生しやすく、知識の継承が難しいソフトウェア開発プロジェクトのマネジメントにおいて、プロジェクトマネジャーは、日々の変動を定量的に詳細な粒度で可視化するための仕組み（システム）と、変動の原因に応じた対策のパターンとして形式知化された過去の経験や知識を活用することによって、プロジェクトを円滑にコントロールすることが可能である。この際に活用される知識がどのような要素から構成されるかについては、知識構造モデルとして明確化された。

図 8-1 の変動の可視化手法や図 8-2 の変動に対するコントロール手法によって、プロジェクトの変動の定量的な把握方法とコントロール方法を、形式知として活用、継承することが可能となる。さらに、変動に対するマネジメトの知識を一般化し、他の分野への適用も含めた知識継承に活用可能な知識構造モデル（図 8-3 及び図 8-4）によって、プロジェクトの変動に対するプロジェクトマネジャーの知識を、対象となるプロジェクトに対する変動の監視、変動へのコントロールと、そのための管理要素に関する知識及びプロジェクトを取り巻く組織、文化等を含む実践知に分類し、知識間の動的な関係の枠組みを示した。

8.2. 理論的な含意

本研究の理論的な含意は、暗黙知となっていることが多いソフトウェア開発のプロジェクトマネジメントの知識継承をテーマとし、変動可視化の自動化を中心とした詳細な管理粒度のデータを利用する定量的マネジメント手法と、パターン化による過去の経験の形式知化を組み合わせることで、過去のプロジェクトの知識活用や知識継承を促進可能であることを示し、新たな研究に貢献するものである。

定量的マネジメントの研究としての理論的な側面としては、従来の研究は、計画の精度を向上させるための規模見積り、工数見積り及び品質評価等のメトリクスの精度向上が主体であったのに対し、本研究は、規模、進捗、コスト等の変動への対処方法に焦点を当てたものであり、変動の監視や制御におけるメトリクスの活用と、知識の形式知化の組合せが有効であることを示した新たな観点での研究である。

知識科学の観点では、従来の研究が、知識継承のプロセスとしてのモデル化の観点や、知識の性質による分類方法等が研究対象であったのに対して、本研究では、プロジェクトマネジメントの知識がいかなる要素から構成されているかと、知識要素間の関連性を明らかにするためのモデルを提案したものである。特に、制御対象としてのプロジェクトに対する変動の監視と制御の観点で知識要素を構造化しモデル化することによって、具体的なマネジメントプロセスと一体として知識のシステム化や形式知化を進めるための枠組みを提示した。

提案した知識構造モデルの特徴は、制御対象としてのプロジェクトに対する監視（モニタリング）と制御（コントロール）及び、プロジェクトの状態を表す管理要素とそれを介した変動管理、及び背景知識となる実践知から成る知識レイヤーによって知識領域を分類し、各知識レイヤーや要素間の関係を、マネジメントのプロセスに対応させて示した点である。これによって、プロジェクトマネジメントの各種知識を、マネジメントの場面や用途に応じて明確にすることが容易となるため、マネジメント手法との組合せによる知識の蓄積・活用方法の研究や、知識を補完するためのシステム化とその実現方法の研究などの、知識継承の枠組みとしての研究の発展に貢献することが期待される。

8.3. 実務的な含意

実務的な観点での含意は、変動が発生しやすく定量化が難しいソフトウェア開発のプロジェクトマネジメントの課題を解決するため、プロジェクトの変動への対処に着目したマネジメント手法と、マネジメントの際に必要となる知識要素を明確にする知識構造モデルを活用することで、組織内のプロジェクト経験を形式知化し、過去の知識やデータを活用してプロジェクトの成功率を高めることを可能としたことである。

変動のマネジメント手法としては、ソフトウェア開発プロジェクトにける見積りや計画

の精度を向上させるだけでは防止することができない、規模、進捗、コストの変動に対し、日次での変動把握手法と過去の知識を形式知化したコントロール手法によって、早期の異常検知や予防的な対策を行い、計画と実績の差異を少なくするための円滑なマネジメントを可能とした。

EVM の適用方法に関する従来の研究は、出来高（進捗）とコストを一体として定量化することでプロジェクト全体の傾向を俯瞰的に把握する手法としての研究であったが、本研究は、詳細な粒度かつ短い管理サイクルで可視化することと、進捗や工数と合わせてその基礎となる規模の変動も合わせて可視化し、進捗やコストに影響を及ぼす生産性や習熟度などのメトリクスも一体として管理することによって、EVM を、日次での詳細な変動把握と原因分析が可能な手法（デイリーEVM 手法）として拡張した。

変動に対するコントロール方法については、再見積りなどの対処に関するプロジェクトマネジャーの暗黙知となっていた知識を、変動の原因別にパターン化して形式知とすることで、過去の知識の活用や継承を可能とした。

詳細な WBS の粒度での日々の変動監視を実現する上で、阻害要因となっていた運用負荷等の問題を考慮し、進捗率の計上や工数計画作成の自動化と過去の人間のノウハウのパターン化によって、運用負荷の軽減と精度の向上を両立した。進捗率の計上については、工数実績データを活用した進捗計上ルールの設定によって、進捗率計上の自動化を実現した。詳細な WBS 別の基準工数をベースラインとし、主として工数実績データから進捗率を計上できるよう、進捗計上のルールを設計しシステム化することによって、運用負荷を増やすことなく、日次での変動の可視化を実現した。この仕組みにより、早期の変動の把握と、原因に応じた早期の対策による円滑なプロジェクトのコントロールが可能となった。

また、工数計画の一括修正を可能とすることにより、実績データに基づいてパラメータを見直し、計画の精度を高めることを可能とした。このしくみは、見積りの精度を高めることには限界があることから、日次の詳細な実績データを元に、早期に補正を行うことによって計画の精度を高めることを可能としたという点で、実務的な意義があると言える。

コントロールに関する知識の活用の視点では、プロジェクトマネジャーの経験に依存していた再見積りなどの手法をパターン化することによって形式知化し、ステークホルダとの合意形成を含むマネジメントプロセスのモデルを提案した。コントロールに関する知識領域については、個人の暗黙知に依存する部分が大きいが、この手法により、変動の原因に応じて過去の知識を活用することによるコントロールが可能となった。以上の変動可視化とコントロールの知識の形式知化を中心としたマネジメント手法により、異常の早期察知、問題の原因分析やコスト予測等を通じた的確なコントロールが可能となった。

プロジェクトマネジメントの知識継承の観点としては、PMBOK や ISO21500 などの知識体系において、計画、実行、監視、制御等の作業プロセスの種類を分類する観点で体系化が進められてきた。しかし、それらの知識体系は、何を行うべきかという What の種類を分類するに止まり、どのようにプロジェクトの状況を監視、コントロールするのかという、

How や Why に関する知識を蓄積及び活用する上では、利用しにくかった。また、変動に対する原因究明や対処方法については、プロジェクトマネジャーの暗黙知の領域と見なされていた部分が多く、知識の体系化が進んでいなかった。本研究では、プロジェクトを監視及び制御する観点で知識を階層化し、管理要素の概念を導入し、知識階層間の動的な関係性を表現することによって、知識構造モデルとして体系化した。これによって、プロジェクトマネジメントのプロセスに対応させて、どの知識を組み合わせて活用すれば良いかが明確になる。また、蓄積すべき知識領域の特定や、領域に応じた知識の形式知化の方法の検討等に活用することができる。このモデルの活用効果を、プロジェクトの変動を可視化する手法を適用した事例と、プロジェクトマネジメントへの AI 技術適用方法の考察によって検証することにより、プロジェクトマネジメントの知識のパターン化による蓄積や、システム化及び AI 技術適用などの検討における枠組みとして有効であることを示した。

8.4. 本研究の評価方法及び適用範囲に関する限界

本研究は、ソフトウェア開発のプロジェクトマネジメントの知識継承に視点を置き、マネジメント手法と知識の形式知化を組み合わせることで、過去の知識を活用したマネジメントの実行やそのための知識継承の促進に貢献するものである。

8.4.1. 効果の評価方法について

研究のアプローチとしては、デザインサイエンスの方法論によって、手法の提案と適用実績による検証を行った。実際のプロジェクトでの適用結果によって、早期の問題把握やデータ活用による原因分析、及び原因に応じた対策の実施や、再見積り手法による考慮漏れの少ない計画修正等による、変動の抑制の効果を検証した。定量的な効果は、本手法を適用しなかった場合との比較シミュレーション等によって示した。本手法は、同一組織内の 7 プロジェクト以上での適用実績があるが、全く同じ特性のプロジェクトは存在しないため、効果を統計的な定量データ等で示すことが難しい。本手法を継続的に適用してゆくことで、知識継承の効果を含む、定性的な効果を積み重ねる必要がある。

8.4.2. 変動マネジメント手法の適用範囲や課題と対策について

提案した変動マネジメント手法は、変動監視の自動化と過去の経験やノウハウのパターン化を組み合わせ、マネジメントプロセスに組み込むことによって、暗黙知となっていたプロジェクトマネジャーの知識の活用や継承を促進するものである。ただし、想定していないパターンの変動への対応等が発生した場合には、マネジメント手法の中に組み込まれた知識の背景や基本原理等を理解した上で本手法を応用し、原因分析やコントロールのための対策を実行することが必要となる。例えば、本手法の基礎となっている管理要素や知

識要素間の関係や原理を理解することが重要である。経験の浅いプロジェクトマネジャーがこのような知識の習得を効果的に進めるためには、本手法を初めて適用する際に、熟練者による指導や OJT（On the Job Training）等を合わせてマネジメントを実施することが有効であり、この際に、知識構造モデルを利用することによって、必要な知識を整理しながら移転することが期待される。

また、本手法は、見積りや計画の誤差は避けられないという考え方に基づき、詳細な管理単位での進捗やコストの変動を、短いサイクルで把握し、見積りや計画の精度を補正しながら、早い段階でプロジェクトの変動が少なくなるようにコントロールするという考え方に基づいたものである。したがって、正確な工数実績データ等を収集し、ステークホルダの間でプロジェクトの状況を定量的に共有することが必要であり、発注者と受託者の間でのプロジェクトの推進方法の事前合意や、データを活用したマネジメント方法を組織内で定着させる取り組みと合わせて、提案した手法を適用することが重要である。

次に、変動マネジメント手法に含まれる見積りや進捗率などのデータの精度に関する課題について述べる。例えば、工数計画の作成や工数の再見積方式等において、実績データに経年変化が発生した場合、古い実績データを適用することによる見積り誤差等が発生することも予想される。このような誤差を少なくするためには、統計処理を行う際の母集団を、開発技術の違いなどのプロジェクトの特性によって分類することや、経年変化を考慮した移動平均等の統計処理方法を採用することによってデータの経年変化に適応させるなどの施策が必要となると考える。また、進捗率の自動計上の仕組みにおいて、自動計算された結果と実態との誤差が発生する可能性がある。これらの問題については、提案した手法においては、第 4 章や第 5 章で述べたように、初期の開発作業の実績データを用いて生産性、スキル係数、習熟度等を見直し、計画を修正することが容易な仕組みとなっているため、実績データを用いて各パラメータを補正し、精度を高めることが可能となっている。

なお、本研究は、企業内のシステムの開発などを、ウォーターフォール型の開発プロセスモデルを適用して推進する場合を想定したものであるが、どのような開発手法を適用する場合でも、変動を短期間に把握して早期に対策を講ずるための方法を確立しておくことは重要である。アジャイル開発を適用するプロジェクトにおいても、イテレーション単位でのタスク実行のための工数の変動把握は必要であり、WBS の粒度や変動可視化のためのデータの収集方法、及び管理サイクル等を見直すことで、提案した変動マネジメント手法の考え方を応用することが可能と考えられる。

8.4.3. 知識構造モデルの適用分野について

提案した知識構造モデルについては、知識継承に関する先行研究の多くが、知識の共有、活用、継承のプロセスやメカニズムのモデル化を行ったものであったのに対して、本研究では、継承すべき知識の特定や、知識の種類に応じた適切な知識継承の方法の選択の枠組

みとして活用可能な知識構造モデルを提案した。具体的には、プロジェクトの変動に対する監視と制御の観点で具体的な知識を構造化し、形式知や暗黙知の観点で分類しやすい形式でモデル化した。

本モデルは、ウォーターフォール型のシステム開発のプロジェクトを想定したものであるため、他の分野や、アジャイル開発モデルを適用したプロジェクトの場合には、提案したモデルの適用範囲を検証する必要があると考える。その場合でも、プロジェクトを制御対象として捉えたときの知識の階層化や、マネジメントプロセスと知識領域の関係の定義、知識要素間の関係性の定義などの考え方を応用することが可能と考える。

8.5. 今後の展望

近年、ICT の活用目的が、既存業務の効率向上から、新たなビジネスモデルの創出等への活用に移ってきている。このため、プロジェクトマネジメントの知識の蓄積や継承及びそれを支えるマネジメント手法が一層重要性を増してくると考える。また、プロジェクトの初期段階で要件や仕様を確定することは、より難しくなり、ソフトウェアの規模やスコープが変動することが多くなると予想される。そのようなケースにおける変動の把握や対処の方法について、分野に合わせた手法の検討が必要となると考える。

本研究で提案した変動の可視化手法と知識構造モデルは、企業内の IT システム開発等を前提として提案したものであるものであるが、既存の知識体系である PMBOK などとの対応関係を検証することによって、汎用性が保たれていると考える。このため、他の分野での知識構造モデルの研究への発展が期待される。例えば、知識構造モデルの中の「管理要素」の種類等については、分野に応じて追加や削除を行うなどの見直しは必要となるが、プロジェクトの状態を把握しコントロールを行うためには、データや知識を活用するための指標としての管理要素を明確にし、過去のノウハウ等の形式知化を進めることが重要であり、本研究の考え方を応用することが可能である。ただし、研究開発や、共創プロジェクト等においては、進捗やコストだけでなく、スコープや実現方法自体が大きく変わることもあるため、例えば、企画やアイデア創出、プロトタイプ検証、構築等の各段階に応じたマネジメント方法や知識構造モデルが必要となると考える。

今後、IoT の拡大や AI 技術等の適用を伴った ICT がデジタルトランスフォーメーションを牽引してゆくと見込まれており、一層複雑なプロジェクトのマネジメントが要求されると考えられる。その際に必要となる知識の蓄積や活用及び継承のため、本研究で提案したマネジメント手法と知識の形式知化の組み合わせ、及び知識構造モデルの枠組みが、さらなる研究の発展に寄与することが期待される。

参考文献

Albrecht, A. J. (1979). “Measuring application development productivity,” *Proceedings of the IBM application development symposium*, 14–17 Oct 1979. IBM Corporation, Monterey, CA, 8-92.

Anbari, F. (2003). "Earned value method and extensions,” *Project Manage Jourrnal*. 34(4): 12-23.

Berlin, S., Raz T., Glezer C., and Zviran M. (2009). "Comparison of estimation methods of cost and duration in IT projects". *Information and Software Technology*, 51: 738-748.

Boehm, B. W. (1981). *Software Engineering Economics.* Englewood Cliffs, N.J., Prentice Hall.

Boehm, B. W. (1984). “Software Engineering Economics”. *IEEE Transactions on Software Engineering*, 10(1): 4-21.

Boehm, B. W., Abts, C., Brown, A.W., Chulani S., Clark B.K., Horowits E., Madachy R., Reifer, D. and Steece B. (2000). *Software Cost Estimation with COCOMO II*. Prentice Hall.

Briand, L. C., Wieczorek, I. (2002). *Resource modeling in software engineering. In: Marciniak J. J. Encyclopedia of software engineering, 2nd edn.* Wiley, New York.

Brooks, F. (1995). *The Mythical Man-Month: essays on software engineering*, Addison Wesley.（＝2002，滝沢 徹 (翻訳)，牧野 祐子 (翻訳)，『人月の神話―狼人間を撃つ銀の弾はない（新装版）』．ピアソン・エデュケーション.）

Callegari, D. A., (2003). “Project management and software development processes integrating RUP and PMBOK,” *Proceedings of the 2007 International Conference on Systems Engineering and Modeling*. IEEE.

Callegari, D. A. and Batos R. M. (2012). "Project Management and Software Development Processes: Integrating PMBOK and OPEN". *World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering*, 6(2): 182-190.

Chidamber, S., Kemerer, C. (1994). "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering* 20: 476-93.

Cicmil, S. and Hodgson, D. (2006). “NEW POSSIBILITIES FOR PROJECT MANAGEMENT THEORY: A CRITICAL ENGAGEMENT,”. *Project Management journal*; 37(3): 111-122.

Cohn, M. (2004). *User stories applied: for agile software development.* Addison-Wesley Professional, Boston, MA.

Cohn, M. (2005). *Agile estimating and planning*. Prentice-Hall, Upper Saddle River, NJ.

Cooper, R., Kaplan, R. S., Maisel, L. S., Morrissery, E. and Oehm, R. M. (1993). *Implementing Actibity-Based Cost Management: Moving from Analysis to Action: Implementation Experiences at Eight Companies*, Inst of Management Accountants. KPMG.（＝1995，KPMG センチュリー監査法人（訳）『ABC マネジメント革命―米国企業を再生させたコスト管理手法』，日本経済新聞社.）

Corazza, A., Martino, S.D., Ferrucci, F., Gravino, C. and Mendes, E. (2011). “Investigating the use of support vector regression for web effort estimation,” *Empirical Software Engineering*, 16, 211-243.

Davenport, T. H. and Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press, （＝2000，梅本勝博（訳），『ワーキング・ナレッジ―「知」を活かす経営』，生産性出版）

De Long, D. W. (2004). *Lost Knowledge: Confronting the Threat of an Aging Workforce.* Oxford University Press.

Demirors, P. E. (2013). *Applying EVM in A Software Company-revised, Benefits and Difficulties*, 2013.

Elshaer, R. (2013). “Impact of sensitivity information on the prediction of project's duration using earned schedule method”, *International Journal of Project Management*, 31 (4): 579 - 588.

Emran, A.A., Jadallah, A., Paikari, E. (2010). "Application of Re-estimation in Re-planning of Software Product Releases," *International Conference on Software Process*, July 08-09 In Lecture Notes in Computer Science 6195, 260-272.

Fleming, Q.W., and Koppelman, J. M. (2010). *Earned value project management (Forth Edition).* Project Management Institute. （＝2004，PMI 東京（日本）支部（監訳）．『アーンド・バリューによるプロジェクトマネジメント』．日本能率協会マネジメントセンター.）

Fleming, Q. and Koppelman, J. (2003). "What's your project's real price tag?,” *Harvard Business Review* ; 81(9): 20-21.

Garmus, D. and Heron, D. (2000). *Function Point Analysis: Measurement Practices for Successful Software Projects 1st Edition.* （＝2002，児玉公信（監訳），『ファンクションポイントの計測と分析』．ピアソン・エデュケーション.）

Garmus, D. and Herron, D. (1996). *Measuring the Software Process: A Practical Guide to Function Measurements*, Prentice Hall (1996). （＝1999，阪田勇夫（訳），「ソフトウェア機能性の計測：ファンクションポイント技法の実践的入門」，トッパン.）

Ghosh, S. (2015). “Enhance PMBOK by comparing it with P2M, ICB, PRINCE2, APM and Scrum Project Management Standards,” PM World Journal, 15(1): 1-77.

Halstead, M. H. (1977). *Elements of software science.* Operating and programming systems series. Elsevier Science, New York.

Hawk, D. L. (1999). “Learning Project Management,” International Project Management Journal.

Hevner, A. R., March, S., T., Park, J. & Ram, S. (2004). “Design Science Research in Information Systems” *MIS Quarterly*, 28(1): 75-105.

Hidding G. J. (2014). “Reducing I.T. Project Management Failures Early Empirical Results”, IEEE, 2014 47th Hawaii International Conference on System Science, 4305-4314.

Horner, R. B. and Yong, W. S. (2006). “Searching for knowledge in the PMBOK guide,” *Project*

Management Journal. 37(2): 11-26.

Huang, S.J., Chiu, N.H. and Liu, Y.J. (2008). "A comparative evaluation on the accuracies of software effort estimates from clustered data," *Information and Software Technology*, 50(9–10): 879–888.

Huang, S.J. and Chiu, N.H. (2009). "Applying fuzzy neural network to estimate software development effort," *Application Intell*, 30: 73-83.

Idri, A., Amazal, F. and Abran, A. (2015). "Analogy-based software development effort estimation: a systematic mapping and review," *Information and Software Technology*, 58: 206-230.

IFPUG (2010). *Function Point Counting Practices Manual Release 4.3.1*. International Function Point Users Group.

IPA (2005). 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,『ソフトウェア開発見積りガイドブック』, オーム社.

IPA (2006).『経営者が参画する要求品質の確保』. 情報処理推進機構ソフトウェアエンジニアリングセンター. 株式会社オーム社.

IPA (2006).『経営者が参画する要求品質の確保　～超上流から攻める IT 化の勘どころ～ 第 2 版』, 情報処理推進機構ソフトウェアエンジニアリングセンター. 株式会社オーム社.

IPA (2008). 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,『定量的品質予測のススメ―IT システム開発における品質予測の実践的アプローチ (SEC BOOKS)』, オーム社.

IPA (2018). 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（偏）.『ソフトウェア開発データ白書 2017-2018』. 独立行政法人情報処理推進機構（IPA）社会基盤センター.

IPA (2018). 情報処理推進機構ソフトウェア・エンジニアリング・センター（SEC）.『IT プロジェクトの見える化～総集編～』, 日経 BP 社.

ISO/IEC (2005). ISO/IEC24570 : 2005 . *Software engineering － NESMA functional size measurement.*

ISO21500: 2012. *Guidance on project management*.

JIS X 0135-1 : 2010『ソフトウェア測定 - 機能規模測定 - 』

JIS X 0142 : 2015『ソフトウェア技術－機能規模測定－IFPUG 機能規模測定手法（IFPUG4.1 版未調整ファンクションポイント）計測マニュアル』, 日本規格協会.

Jones, C. (1996). *Applied software measurement: assuring productivity and quality, 2nd edn*. McGraw-Hill Osborne Media, New York.

Jones, C. (2004). "Software Project Management Practices: Failure versus Success," *CROSS TALK the Journal of Defense Software Engineering*. 17(10): 5-9.

Jones, C. (2006). "Social and Technical Reasons for Software Project Failures,' *CROSS TALK the Journal of Defense Software Engineering*. 19(6), 4-9.

Jones, C. (2007). *Estimating Software Costs Bringing Realism to Estimating Second Edition.* The McGraw Hill.（＝2009 年，富野 壽（翻訳），岩尾 俊二（翻訳），『ソフトウェア見積もりのすべて －現実に即した規模・品質・工数・工期の予測－ 第 2 版』．構造計画研究所.）

Jones, C. (2008). *Applied Software Measurement: Global Analysis of Productivity and Quality Third Edition*, McGraw-Hill.（＝2010，富野春・小坂恭一（監訳），『ソフトウェア開発の定量化手法 第 3 版』，共立出版株式会社.）

Jones, C. (2013). "Function points as a universal software metric," *Newsletter, ACM SIGSOFT Software Engineering Notes*, 38(4): 1-27.

Jrgensen, M. (2004). "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, 70(12): 37-60.

JUAS (2016). 社団法人 日本情報システム・ユーザー協会，『企業 IT 動向調査 2016』. http://www.juas.or.jp/library/research_rpt/it_trend/（2018 年 11 月 22 日アクセス）

JUAS (2018). 社団法人 日本情報システム・ユーザー協会，「2018 年版 ユーザー企業ソフトウェアメトリックス調査」. http://www.juas.or.jp/library/research_rpt/it_trend/（2018 年 11 月 22 日アクセス）

Karner, G. (1993). "Resource Estimation for Objectory Projects," *Technical report, Objective systems* SF AB, Kista, Sweden.

Kerzner, H. (2009). *Project management: a systems approach to planning, scheduling, and controlling (10th Ed.)*. John Wiley & Sons, Inc.

Kinsella, Kinsella S., Steven, M. (2002). "Activity-based costing: Does it Warrant Inclusion in A Guide to the Project Management Body of Knowledge (PMBOK Guide) ?," *Project Management Journal*, 33 (2): 49-56.

Kirmani, M. (2017). "Software Effort Estimation in Early Stages of Software Development: A Review," *International Journal of Advanced Research in Computer Science*, 8(5): 1155-1159.

Kocaguneli, E, Menzies, T. and Keung, J.W. (2012). "On the value of ensemble effort estimation," *IEEE Transactions on Software Engineering*, 38(6): 1403-1416.

Kumar, K.V., Ravi, V., Carr, M. and Kiran, N.R. (2008). "Software development cost estimation using wavelet neural networks," *Journal of Systems and Software*. 81(11):1853-1867.

Lipke, W. (2003). "Schedule is different,' *The Measurable News* 2003 (March), 31-4.

Lipke, W., Zwikael, O., Henderson, K. and Anbari, R. (2009). "Prediction of project outcome," *International Journal of Project Management*, 27(4): 400-407.

Liu, Q., Qin W., Mintram, R. and Ross, M. (2008). "Evaluation of preliminary data analysis framework in software cost estimation based on ISBSG R9 data," *Software Quality Journal*, 16: 411-458.

Mah, V. (2012). *Measuring Progress of Scrum-based Software Project*, 2630-7621-1-PB

Marciniak, J. J. (1994). *Encyclopedia of Software Engineering*, Wiley-Interscience. （＝1998，片山卓也（翻訳），土居範久（翻訳），『ソフトウェア工学大事典』．朝倉書店，1420-1430.）

McCabe, T. (1976). "A Complexity Measure,". *IEEE Transations on Software Engineering* 4: 308-320.

Mendes, E. (2014). *Practitioner's Knowledge Representation A Pathway to Improve Software Effort Estimation*. Springer-Verlag Berlin Heidelberg.

Milton, N. (2005). *Knowledge Management: For Teams and Projects*, Chandos Publishing.（＝2009, 梅本勝博・石村弘子（監訳）.『プロジェクト・ナレッジ・マネジメント』, 生産性出版.

Minku, L.L. and Yao, X. (2013). “Software effort estimation as a multiobjective learning problem,” *ACM Transactions Software Engineering Methodology* 22(4): 35.1-35.32.

Mittal. A., Parkash, K., and Mittal, H. (2010). “Software cost estimation using fuzzy logic,” *SIGSOFT Software Engineering Notes*, 35(1): 1‑7.

Muzaffar, Z. and Ahmed, M.A. (2010). “Software development effort prediction: a study on the factors impacting the accuracy of fuzzy logic systems,” *Infomation Software Technology*, 52(1): 92‑109.

Nonaka, I. and Takeuchi, H. (1995). *The Knowledge-Creating Company.* Oxford University Press.

Oliveira, A.L.I. (2006). “Estimation of software project effort with support vector regression,” *Neurocomputing*, 69(13‑15): 1749‑1753.

Oliveira, A.L., Braga, P.L. and Lima, R.M. and Cornlio, M.L. (2010). “GA based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation,” *Information and Software Technolgy*, 52(11): 1155‑1166.

Oshima, T. and Maruyama, T. (2016). “Project Management Approach using the Visualization of Changing Software Size,” *Proceedings of the 10th International Conference on Project Management (ProMAC2016)*, 419-424.

Oshima, T and Maruyama, T. (2017). “The Method for Visualizing Variance of Software Development Project in Real Time,” *Proceedings of the 11th International Conference on Project Management (ProMAC2017)*, 401-416.

Pajares, J. (2011). “An extension of the EVM analysis for project monitoring,” *International Journal of Project Management*, 29(5): 615-621.

PMAJ (2014).『改訂 3 版 P2M プログラム & プロジェクトマネジメント標準ガイドブック（Program & Project Management for Enterprise Innovation）』．日本能率協会マネジメントセンター.

PMI (2013). Project Management Institute. *A Guide to the Project Management Body of Knowledge: PMBOK Guide*. Project Management Institute. (Project Management Institute (2015).（＝2013,『プロジェクトマネジメント知識体系ガイド（PMBOK ガイド）第 5 版』．Project

Management Institute.）

PMI (2015). "Capturing the Value of Project Management through Knowledge Transfer," *Pulse of the Profession In-Depth Report*. Project Management Institute, Inc.

PMI (2017). Project Management Institute. *A Guide to the Project Management Body of Knowledge: PMBOK Guide Sixth Edition*. Project Management Institute. （＝2017,『プロジェクトマネジメント知識体系ガイド（PMBOK ガイド）第 6 版』 Project Management Institute.）

Polanyi, M. (1980). 高橋勇夫（翻訳）『暗黙知の次元』ちくま学芸文庫.

Polikar, R. (2006). “Ensemble based systems in decision making,” *IEEE Circuits Systems Magazine*, 6(3): 21-45.

Putnam, L.H. (1978). "A general empirical solution to the macro software sizing and estimating problem," *IEEE Transactions Software Engineering*, 4(4): 345-361.

Putnam, L.H. and Myers (1992). W., *Measures for Excellence*, Prentice Hall Yourdon Press, Englewood Cliffs, NJ. （＝1995, 研野 和人（訳),『プロジェクトの見積りと管理のポイント』, 共立出版.）

Raz, Tzvi R. and Dan E. (1999).” Activity based costing for projects’, International Journal of Project 201 International Association of P2M NII-Electronic Library Service Management, 17 (1): 61-67.

SWEBOK (2004). *SWEBOK Guide 2004.* IEEE Computor Society.

Trendowicz, A. (2013). *Software Cost Estimation, Benchmarking, and Risk Assessment, The Software Decision-Makers' Guide to Predictable Software Development*, Springer-Verlag Berlin Heidelberg.

Trendowicz, A. and Jeffery, R. (2014). *Software Project Effort Estimation, Foundations and Best Practice Guidelines for Success*. Springer International Publishing.

TSO (2009). The Stationery Office. *Managing Successful Projects with PRINCE2*. The Stationery Office.

Uchihira, N. (2005). "Stage Gate Analysis in Business-Academia Collaborative Project," *In Proceedings of PICMET2005*.

Van Aken, J., E. (2005). "Management research as a design science: Articulating the research products of mode 2 knowledge production in management". *British Journal of Management*, 16(1): 19-36.

Vandevoorde, S. and Vanhoucke, M. (2006). “A comparison of different project duration forecasting methods using earned value metrics,” *International Journal of Project Management*, 24 (4): 289-302.

Yucalar, F., Kilinc, D., Borandag, E. and Ozcift, A. (2016). "Regression analisys based software effort estimationmethod," *International Software Engineering*, 26(05): 807-826.

Zack, M.H. (1999). “Managing Codified Knowledge,” *Sloan Management Review*.

阿萬裕久・山下裕他（2010）.「整数計画法を用いた重点レビュー対象モジュールの選択」. コンピュータソフトウェア，27(4): 240-245.

阿萬裕久・野中誠・水野修（2011）.「ソフトウェアメトリクスとデータ分析の基礎」，日本ソフトウェア科学会，コンピュータソフトウェア，28(3), 12-28, 2011-07-26.

安宅和人（2015）.「人工知能はビジネスをどう変えるか」，DIAMOND ハーバード・ビジネス・レビュー2015 年 11 月号，ダイヤモンド社，47.

内田吉宣・鮫島正樹・藤波努・星幸雄・初田賢司・建部清美（2010）.「プロジェクトマネジメントにおける経験知抽出方法」,『プロジェクトマネジメント学会誌』, 12(4): 27-32.

内田吉宣・坂上慶子・酒井俊永・鴈丸明仁（2013）.「マネジメントにおける実践的知恵養成のためのケースメソッド」.『プロジェクトマネジメント学会誌』，15(2): 9-13.

内田吉宣(2016).「開発プロジェクトにおけるリスク知識の組織内知識継承マネジメント」，博士論文，北陸先端科学技術大学院大学.

内平直志（2010）.「研究開発プロジェクトマネジメントの知識継承」，博士論文，**北陸先端科学技術大学院大学**，**博知第 124 号**.

大島丈史・丸山富子（2016）.「ソフトウェア規模の変動可視化によるプロジェクト管理手法」，プロジェクトマネジメント学会，『プロジェクトマネジメント学会誌』， 19 (1): 26-31.

大島丈史・丸山富子（2016）.「変動要因を考慮した工数見積方式に基づくプロジェクト管理手法」,『プロジェクトマネジメント学会 2016 年秋季研究発表大会予稿集』, 201-206.

大島丈史・丸山富子（2017）.「WBS 単位の詳細実績データ活用によるソフトウェア開発の変動管理手法」,『プロジェクトマネジメント学会 2017 年度春季研究発表大会予稿集』, 289-294.

大島丈史・内平直志（2018）.「開発プロジェクトの変動マネジメントに関する知識構造モデルの活用」,『プロジェクトマネジメント学会 2017 年度春季研究発表大会予稿集』, 18-23.

大島丈史・内平直志（2018）.「プロジェクトマネジメントへの AI 活用の知識分類モデル －IT 企業における AI 適用方策の研究－」,『Journal of International Association of P2M, 国際 P2M 学会誌 2018』，13(1):121-141.

岡田公治（2017）.「プロジェクト規模の多様性を考慮したプロジェクトマネジメント行動ルールの機械学習可能性に関する検討」,『プロジェクトマネジメント学会，2017 年度春季研究発表大会予稿集』，94-101.

河村智行・高野研一（2018）.「ソフトウェア開発プロジェクトの成否予測に関する研究の調査」，慶應義塾大学大学院 システムデザイン・マネジメント研究科.

神林友和（2018）.「AI 技術を活用した高リスクプロジェクト予兆検知手法の一提案」,『プロジェクトマネジメント学会 2018 年度春季研究発表大会予稿集』，423-428.

河野善彌・陳慧（2010）.「ソフトウェア開発工程の定量的特性」， 情報処理学会研究報告.

木野泰信（2011）.「ソフトウェア開発プロジェクトの見える化」.『プロジェクトマネジメント学会誌』，31(121): 65-70.

齊藤 毅・鈴木 研一（2013）.「EVM を用いたコストマネジメントの限界と ABC の適用」，『P2M 学会研究発表大会予稿集』，Autamn (0): 182-202.

情報処理振興事業協会（2003）.『EVM 活用型プロジェクト・マネジメント導入ガイドライン』. http://www.meti.go.jp/policy/it_policy/tyoutatu/evm-guideline.pdf （2018 年 11 月 26 日アクセス）.

総務省（2016）.『ICT の進化が雇用と働き方に及ぼす影響に関する調査研究』（平成 28 年）

鈴木研一（2014）.「ロジェクトのマネジメントコントロール研究の視点」，経営論集.

大峡光晴・河野綾子・鎌田瞬・後田修治・高木伸・山出康介・羽根孝泰（2017）.「AI を活用した不採算プロジェクトの予兆検知」，『プロジェクトマネジメント学会誌』，19(4): 15-20.

建部清美・関哲朗（2006）.「CCPM 法適用プロジェクトにおける EVM 導入の一考察」，『プロジェクトマネジメント学会誌』，8(1): 29-32.

中村正伸（2013）.『EVM を用いた予実差異の原因分析の可能性－実行段階での資本予算の管理のために』. 原価計算研究，37(2): 21-32.

日経 BP 社（2018）.「IT プロジェクト実態調査 2018」. 日経コンピュータ 2018 年 3 月 1 日号，26-39.

野中郁次郎・竹内 弘高（2011）.「賢慮のリーダー」Harvard Business Review， 2011 年 9 月号，10-24.

野中誠（2008）.「ソフトウェア開発コスト予測研究の動向と課題」.『ソフトウェアエンジニアリング最前線』，飯田元，山本絵里子（偏），近代科学社，33-40.

箱嶋俊哉（2007）.「EVM とクリティカル・パスの視点」，『プロジェクトマネジメント学会誌』，9(2): 13-17.

初田賢司・原田晃・大野治（2002）.「ソフトウェア開発プロジェクトにおける見積技術」，『プロジェクトマネジメント学会誌』，4(4): 14-18.

東岳人・久米田暁文・角田文広・小室睦・菅沼弘（2003）.「ソフトウェア開発における定量的プロジェクト管理」，『プロジェクトマネジメント学会 2003 年度秋季研究発表大会予稿集』，136-140.

富士通（2009）.「新たな開発規模見積り手法 －ファンクションスケール法の実践－」，『雑誌 FUJITSU』，60 (6): 551-558.

富士通（2013）.「ファンクションスケール法　測定マニュアル　1.1 版」，富士通株式会社，（2016 年 7 月 14 日取得，
http://www.fujitsu.com/jp/solutions/infrastructure/dynamic-infrastructure/sdas/fs/）

富士通（2017）.「AI などの最新テクノロジーを活用し，システム構築・運用業務を変革」，富士通株式会社，(2018 年 3 月 23 日取得，http://pr.fujitsu.com/jp/news/2017/11/28.html).

星幸雄・本田伸一・初田賢司・建部清美（2007）.「ソフトウェア開発におけるアーンドバリュー適用の取り組み」,『プロジェクトマネジメント学会誌』, 9(2): 8-12.

真野俊樹・誉田直美（1993）. 「見積りの方法―ソフトウェア開発における実践的見積り技法」, 日科技連出版社.

前田浩孝（2004）.「成功裏な EVM 適用に向けての考察」.『プロジェクトマネジメント学会 2004 年度春季研究発表大会予稿集』, 93-97.

持田信治（2011）.「EVM 法を使用した生産性とコスト管理について」- 流通科学大学論文集 20(1): 017-028.

米花幸男（2003）.「システム開発における EVM 適用への実用的なアプローチ」,『プロジェクトマネジメント学会 2003 年度秋季研究発表大会予稿集』, 25-30.

渡辺清孝・角田昌嗣・高橋政男（2004）.「EVM におけるクリティカル・チェーン法の適用と有効性」,『プロジェクトマネジメント学会誌』, 6(3):.12-15.

渡辺純・丸山富子（2009）.「プロセス重視によるソフトウェア開発の工業化への取組み」. FUJITSU. 60(6): 567-573.

本論文の骨格となる研究業績リスト

1. 学術誌掲載論文（査読あり）

大島丈史・丸山富子（2017）.「ソフトウェア規模の変動可視化によるプロジェクト管理手法」（査読あり）,『プロジェクトマネジメント学会誌』, 19 (1): 26-31.

【本論文との関係】

変動マネジメントの中の、規模の変動に関する計測及び可視化手法に対応する。

大島丈史・内平直志（2018）.「プロジェクトマネジメントへの AI 活用の知識分類モデル －IT 企業における AI 適用方策の研究－」（査読あり）,『Journal of International Association of P2M, 国際 P2M 学会誌 2018』, 13(1):121-141.

【本論文との関係】

知識構造モデルにおける、知識の分類と活用方法の例に対応する。

2. 国際学会口頭発表論文（査読あり）

Oshima, T. and Maruyama, T. (2016). “Project Management Approach using the Visualization of Changing Software Size,” *Proceedings of the 10th International Conference on Project Management (ProMAC2016)*, 419-424.

【本論文との関係】

変動マネジメント手法における規模変動の可視化手法に対応する。

Oshima, T. and Maruyama, T. (2017). “The Method for Visualizing Variance of Software Development Project in Real Time,” *Proceedings of the 11th International Conference on Project Management (ProMAC2017)*, 401-416.

【本論文との関係】

進捗及び工数の変動の可視化手法に対応する。

3. 国内学会口頭発表論文（査読なし）

大島丈史・丸山富子（2016）.「変動要因を考慮した工数見積方式に基づくプロジェクト管理手法」,『プロジェクトマネジメント学会 2016 年秋季研究発表大会予稿集』, 201-206.

【本論文との関係】

変動へのコントロール手法の中の、工数見積方式に対応する。

大島丈史・内平直志（2018）.「開発プロジェクトの変動マネジメントに関する知識構造モデルの活用」,『プロジェクトマネジメント学会 2017 年度春季研究発表大会予稿集』, 18-23.

【本論文との関係】

変動のマネジメントに関する知識構造モデルの中の、構造モデルに対応する。

4. 学術誌投稿中論文（査読あり）

大島丈史，丸山富子（2018）.「ソフトウェア開発の円滑なコントロールのための日次変動可視化手法」,『プロジェクトマネジメント学会誌』、(査読あり)，ページ数 : 6 ページ,（投稿受理日 2018 年 10 月 30 日）.

【本論文との関係】

変動マネジメント手法の中の、進捗及び工数の可視化のための具体手的なドキュメント類の活用方法に対応する。

以上

謝辞

本研究を進めるにあたり、主指導教員の北陸先端科学技術大学院大学 先端科学技術研究科 知識科学系 教授 内平直志先生には、3 年半に渡り研究の計画段階から最終稿執筆まで、昼夜や休日を問わず親身にご指導いただき、大変お世話になりました。厚く御礼申し上げます。副指導教員の教授 神田陽治先生には、個別ゼミや日常においてアドバイス等をいただき、ありがとうございました。副テーマのご指導をいただきました助教 セラヤ・サモラ・ジャデル・エンリケ先生には、副テーマ研究全般についての親身にご指導いただき、ありがとうございました。

千葉工業大学 教授 下田篤先生には、外部審査委員として専門的視点から大変ご丁寧かつ具体的なアドバイスをいただき、厚く御礼申し上げます。北陸先端科学技術大学院大学先端科学技術研究科の教授 藤波努先生、准教授 白肌邦夫先生、准教授 姜理恵先生には、予備審査や個別ゼミ等において懇切にご指導をいただき、本稿の執筆にあたり多大な示唆をいだきました。厚く御礼申し上げます。

また、変動マネジメント手法の適用に当たってご協力いただきました富士通株式会社及び富士通アプリケーションズ株式会社の関係者の皆様に感謝申し上げます。

その他、北陸先端科学技術大学院大学の多くの先生や、全体ゼミ、研究室での夏合宿や、石川本校での交流会等を通じてご意見等をいただきました、社会人学生の方々に感謝申し上げます。大変良い思い出となっています。

最後に、研究を暖かく見守ってくれた家族に感謝します。