博士学位論文

ソフトウェア開発プロジェクトの工数－工期のマネジメントのための設計情報の写像経路を用いた複雑性評価指標の提案

Complexity Index Focused on Mapping Paths of Design Information for Software Development Project Management

2016 年 3 月

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

榮谷　昭宏

博士学位論文要旨

ソフトウェア開発プロジェクトの工数－工期のマネジメントのための設計情報の写像経路を用いた複雑性評価指標の提案

2016 年 3 月

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

榮谷　昭宏

本論文では，ソフトウェア開発プロジェクトの成功率の低さを課題とし，その改善を目的として，設計情報の写像経路に着目した複雑性指標を提案する．まず初めに 1 章では，ソフトウェア開発プロジェクトの現状について述べる．次に 2 章では，提案するプロジェクトモデルを構築し，提案するマネジメント指標を説明する．そして 3 章では，提案するマネジメント指標とプロジェクトの工数と工期の関係を明らかにする．4 章では，提案する指標をコントロールする方法を紹介する．そして 5 章で，全体をまとめる．以上により，提案する複雑性指標によって，工数と工期もコントロールできることを示す．以下，2 章以降の概要を説明する．

2 章では，ソフトウェア開発プロジェクトの要素間の複雑な関係をシンプルに表すモデルを提示し，その有用性を示す．本モデルでは，設計情報の写像経路に着目したプロジェクトの複雑性指標を用いる．そして，その複雑性指標と生産性（1 人あたりの平均作業日数）及び開発工数が相関することを実証的に検証する．これにより，プロジェクトの生産性向上や工数削減を実現するためには，設計情報を写像する経路の冗長性を低減することが必要であり，そのためには提示したモデルが有用であることを示す．そのモデルとは，公理的設計論に基づいて構築され，ソフトウェア開発プロジェクトにおけるプロセスやプロダクトを構成する要素間の関連の複雑さを表している．その複雑性は，設計情報の写像経路数や各経路の写像の難しさから，プロジェクト全体の写像の難しさを数値化したものである．そして提示したモデルの有効性を示すために，実際に 21 のソフトウェア開発プロジェクトからデータを収集することで検証を行った．その結果，複雑性指標と生産性や開発工数との間に強い相関があることが分かった．これらの検証

結果から，本研究で示したモデルの複雑性指標を低減することはプロジェクトの生産性向上や工数削減に有効であることを確認した．

3 章では，工数と工期の経験モデルが何故累乗関数で表されるのか，そして，その乗数，指数が何故様々な値を取るのか，理論モデルを用いて明らかにし，理論的に工数と工期の関係式を導出する．その理論モデルはプロジェクト内でやり取りされる設計情報の写像経路と，その写像の難しさを変数としたモデルを用いる．本論文の結果，乗数は写像経路数とその写像の難しさに依存した関数で規定され，指数は写像経路数のみに依存した関数で規定されることが分かった．そして，これらの関数を元に理論的に算出した近似式と，実際のプロジェクトデータとを比較し，本理論モデルが実際のプロジェクトの工数と工期の関係を再現していることを実証検証した．以上により，理論モデルによって，工数と工期の経験式と理論式の整合性を確認し，そのメカニズムを明らかにした．

2 章・3 章の結果により，工数と工期が複雑性（写像経路数と写像の難しさ）によって決定されることを明らかにした．最後に 4 章では，複雑性指標を中心としたプロジェクトマネジメントの展望を説明している．これにより，複雑性をマネジメントすることによって，工数と工期をコントロールし易くなると考える．本研究の成果は，プロジェクトの成功率向上に向けたひとつの有用な方策を示したと考える．

Thesis Abstract

Complexity Index Focused on Mapping Paths of Design Information for Software Development Project Management

March 2016

Graduate School of System Design and Management, Keio University

Akihiro Sakaedani

Abstract

This research proposes a complexity index for improving the low success rate of software development projects. In the 1st chapter, I explain problem with software project management. In the 2nd chapter, I develop a software project model and the index for project management, In the 3rd chapter, I develop theoretical formula between project efforts and development schedule, and in the 4th chapter, I show that the index for project management makes it easier to control the project efforts and development schedule.

Software development project members communicate with each other by various means and various subjects. The management for such complexities of the communication has an effect on project quality and development cost. As pointed out by researchers, communication in a project has a real influence on quality and cost. However, researchers have not yet assessed the project complexity. In particular, the correlation with productivity and development cost remains unproved. In this paper, I propose the idea of a complexity index focused on mapping paths of design information in software development project. This research establishes a correlation between the complexity and productivity (average workdays per person), development cost (man-hour). I develop the project model using axiomatic design theory. A large portion of the software development activity

is made up of transforming design information. Simpler mapping paths are better for productivity and development cost. This research proposes the complexity index for mapping paths. I prove the validity of the complexity index by discussing a software development project as a case study.

The 3rd chapter develops a theoretical formula that describes the relationship between man-hour and development schedule. This study theoretically discusses why mathematical models in previous works have exponential relationships, and why their multiplier factors and exponents have various values. My research theory is based on a project model determined by the number of mapping paths for design information, and difficulty of mapping. I show that the multiplier factor is defined by the number of mapping paths and the difficulty of mapping, and exponent is specified by the number of mapping paths. I apply my model to actual development data, and found that my model significantly correlates with actual data.

In the 4th chapter, I explain how to use the complexity index in a project. As a consequence, in this research, I show that the index for project management makes it easier to control the project efforts and development schedule.

目次

【第１章】
緒言

1. 緒言

1-1 はじめに

現状，プロジェクトマネジメントに関する研究が進んでいるにも関わらず，ソフトウェア開発プロジェクトの成功率は一向に改善しない．その要因として近年のプロジェクトの複雑化が挙げられている．

1-1-1 プロジェクトの低い成功率

ソフトウェア開発プロジェクトの失敗の多さについて，Standish Group の Chaos レポート[1]により報告されている．毎年約 70 パーセント程度のソフトウェア開発プロジェクトが失敗している．世の中には多くの開発標準が存在しているにも関わらずその傾向は 10 年以上変わらない．

様々な IT 技術が開発され，社会で活用されているにも関わらず，何故そもそも IT 技術自体を生み出すソフトウェア開発プロジェクトは進化せずに失敗を繰り返しているのだろうか．また，社会の重要なインフラとして機能しているソフトウェアが，このように余裕のない，切羽詰まった状態で開発され，社会に提供されている状況で良いのだろうか．そのような状況で開発されたソフトウェアで，十分な品質が担保できているのだろうか．複数のソフトウェアが連携し，ひとつの社会インフラとなっている現代社会において，ひとつのソフトウェアの故障は重大な社会的トラブルを招く．すなわち，ソフトウェア自体がミッションクリティカルなモノになっている．そのような社会背景にも関わらず，現状のソフトウェアの開発方法論・設計方法（生産技術），プロジェクトマネジメント技術では，プロジェクトの成功率を改善できていないことは，非常に重大な問題ではないだろうか．それは，社会的に非常に大きなリスクを背負っているとは言えないだろうか．

実際その開発標準類（ISO15288, IEEE 1220, EIA632, CMMI, INCOSE Handbook, and PMBOK Guide 等）について，Frank[2]は既存の開発標準類の殆どがプロセス中心であることを指摘している．そして，プロジェクトの成功率の低さ故にプロセス中心の標準の限界説を述べている．現実にはプロジェクトの現場でもプロセス中心のマネジメント技術が主流であり，Program Evaluation and Review Technique (PERT), Critical Path Method(CPM)等のプロジェクトマネジメントツールの利用が主体である[3].

なお，本論文では成功を Quality, Cost, Time で定義する．プロジェクトの成功とプロジェクトマネジメントの成功は異なるとの見解がある[4]．プロジェクトの成功とは顧客のビジネス的な成功なども含んでいる．プロジェクトマネジメントの成功とは計画通りの期間と予算で当初のステークホルダーが要望していた品質のシステムを実現することである．プロジェクトマネジメントの成功はプロジェクトへの成功にも結び付くものである．本研究ではプロジェクトマネジメントの成功，つまり Quality, Cost, Time

の達成を成功と定義付け，以降の議論を進める．

1-1-2複雑化したプロジェクト

Boehm [5]は，プロジェクトが開発するシステムが複雑になってきたことで旧来のプロセスでは立ち行かなくなっていると述べている．また更に，Slegers[6]は，失敗から学ぶことも困難になってきていると述べ，その理由を，システムが複雑になりシステムやその失敗自体を理解することも難しくなってきたためであると述べている．
その失敗の理解に関連して，Boehm[7]は， 急な変更への対応や高信頼性の実現，その他不確実性の対応などがプロジェクトに求められていると述べ，また一方で，ステークホルダーの多様性も増し，The Bank of America Master Net System の事例を用い，互いの要求が相矛盾する場合も多いとも述べている．更にその矛盾を解決するためにはソフトウェア領域とビジネス領域とのトレードオフを理解しなければならないとも述べている．プロジェクトマネジメントとして，単にテクノロジーの領域だけ管理してれば済む訳ではなく，現代のような複雑化したプロジェクトをマネジメントするためには多岐に渡る領域をマネジメントする必要がある．逆にそれ故にプロジェクトが複雑化しているのである．プロジェクトの複雑化とは即ちマネジメントすべき変数が増大し，その変数どうしの相互依存性も増していることを意味する．その結果，マネジメントの難しさが増大し，プロジェクトの成功率も向上していない．

1-2先行研究レビュー

複雑化しているプロジェクトの成功率の低さを改善することが目指すべきゴールである．つまり，当初計画した QUALITY, COST, TIME を達成することが必要である．しかし, 現状では複雑化したプロジェクトをマネジメントするための技術は不十分であり，複雑化したプロジェクトを十分に理解するに至っていない．

1-2-1現代のプロジェクトマネジメント技術の課題

現在のプロジェクトマネジメント技術の主流は前述の通りプロセス中心となっている．しかし，PERT/CPM のようなプロセス中心のマネジメント技術について Eppinger[8]は製品開発で普通に生じている情報交換による手戻りを管理できない．／プロセスの変更に弱い．／プロジェクトで起きている問題が分かり難い．／スケジュールの見積精度が分からない．といった問題を指摘している．

それ以外にも PERT，CPM はプロジェクトの複雑性の 2 次的要因を観察しているのみ[9]，または PERT，CPM はプロジェクトの過去のデータを見ても統計学的にも問題[10], そして PERT, CPM における RCRSP（Resource-Constrained Project Scheduling Problem）を考慮していないにも関わらず, ソフトウェアツールや実践者は PERT, CPM に固執[11]，と述べられている．このようにプロセス中心のマネジメント技術は先行研究においてもその問題点を指摘されている．

そもそも一般的に製品の開発過程とは試行錯誤の繰り返しである([8], [12], [13])．また，その過程は問題解決過程における情報収集過程である[13]，とも述べられている．従って，プロセスは繰り返されて当り前なのである[9]．従って，手戻り表現の難しい PERT/CPM 等ではマネジメントが難しい[8].

1-2-2複雑化したプロジェクトの理解に向けた課題

プロジェクトマネジメントにおいて，理解すべきはプロジェクトの複雑性である[9]．更に, Danilovic[14]は以下のように述べている．プロジェクトは人を組織し仕事を進める．その仕事はコミュニケーションを取りながら，そして同期をとって行われる．そのため，プロジェクトマネージャ（PM）やエンジニアは相互関係を理解しなければならない．そのためにプロジェクト内の情報流通構造を分析・整理可能にするべきと述べている．プロジェクトの複雑性とはプロジェクト内の情報流通構造の複雑性であるという考えである．

複雑化している情報流通構造を分析するための方法の代表格として DSM(Design Structure Matrix)が挙げられる[15][16]．DSM は多くの研究がなされている．作業間の関係性の分析[17][18]だけでなく，プロジェクトの組織構造と製品アーキテクチャーの関係性を情報流通という観点で分析した研究もある[19],[20]．また Lindeman[21]は DSM の概念を更に発展させ，複数の要素の関係性を議論できるように

MDM(Multiple-Domain Matrices)を開発し，情報流通構造の複雑性に関する理解を進めている．しかし，DSM や MDM は，まだ一般的な測定指標は無く，定性的な議論で活用されることが多い．Kaplan[22]の言う通り，測れないものは管理出来ないのである．このように先行研究においてもプロジェクトの複雑さを理解し，管理するには課題が残されている．

1-3 既存の複雑性指標の限界

前述の通り，先行研究でもプロジェクトの複雑性は十分に理解されているとは言い難い．従って，複雑性をコントロールすることも困難な状況であることには変わりない．その結果，プロジェクトの QUALITY，COST，TIME のバランスを取ることも難しく，失敗プロジェクトの低減に繋がっていない．

プロジェクト内の情報流通構造を分析・管理していくためには，前述の通り複雑性を理解しなければならない．複雑性の理解のために DSM 以外にも複雑性に関する研究は多く，Vidal[23]がその概要を以下のように紹介している．先行研究における複雑性の定義は区々で，研究も現在進行形である[24]．現状のプロジェクトマネジメント技術は複雑性の理解まで[24]であるとも指摘され，複雑性の測定とコントロールまで出来ていない．しかし，既設の複雑性指標とその限界について以下のような指摘もある[23]．①スケジュール等，複雑性の 2 次的要因を測定している．そのため，測定指標の信頼性が低い．②モデル化されたプロジェクトの複雑性を測定している．モデル化した時点で，その複雑性指標は主に相互依存性のみを測定している．複雑性はもっと多くの要因によって生じるものであり，相互依存性だけでは偏った指標である．③シャノンの情報理論[25]やシステム思考等，全体論的な測定方法は，その算出方法の難しさや直感的理解が困難である．そのため，実践向きではない．このように大きく 3 つの指摘がなされている．従って，先行研究においても，複雑性をコントロールすることが困難な状況であることには変わりない．その結果，プロジェクトの QUALITY, COST, TIME のバランスを取ることも難しく，失敗プロジェクトの低減に繋がっていない．

そこで，本研究では，上記の課題を解決した複雑性指標を提案し，その指標によって生産性向上，工数低減が図れることを述べる．そして，更にその複雑性指標が工数と工期の関係を決定付ける要素となっていることを示す．最後に，本研究で提案した複雑性指標をプロジェクトマネジメントに活用した場合の活用方法例を述べる．本研究により，プロジェクトの成功率向上に少しでも貢献できれば幸いである．

1-3-1 本研究の目的

プロジェクトの複雑さを理解し，プロジェクトマネジメントを容易化することを目的とする．そのため，プロジェクトの複雑さの要因となる多変数をコントロールしなければならない現状を改善する．本研究ではプロセス中心ではなく，情報中心のプロジェクトモデルを用いることで，多変数ではなく，複雑性という 1 つの変数に簡素化することを提案する．その指標を用いたマネジメントの考え方を説明し，プロジェクトのコントロールを容易にすることを狙う．

1-3-2 本論文の構造

本論文の構造（図 1－1）は，まず初めに 1 章にて，プロジェクト開発の現状および

その問題点を先行研究レビューによりまとめる．そして，先行研究でも指摘された複雑なプロジェクトのマネジメントに関する限界点について述べる．

2 章では，先行研究で指摘された限界を打ち破るため，多変数の関数をひとつの変数による関数に変換するアイディア，すなわち設計情報の写像経路に着目したプロジェクトのモデル（情報写像モデル）及びそのモデルから導出した複雑性指標を説明する．そして，実際のプロジェクトデータを用いて，提案した複雑性指標がプロジェクトの生産性及び工数と相関することを実証的に示す．

次に 3 章では，提案した情報写像モデルと複雑性指標を元に工数と工期の関係性を理論式として導出する．そして，導出した理論式が正しく現実のプロジェクトを表していることを実際のプロジェクトデータを用いて検証する．

最後に 4 章では，複雑性指標をプロジェクトマネジメントにどのように活用するのか，基本的な考え方を述べる．

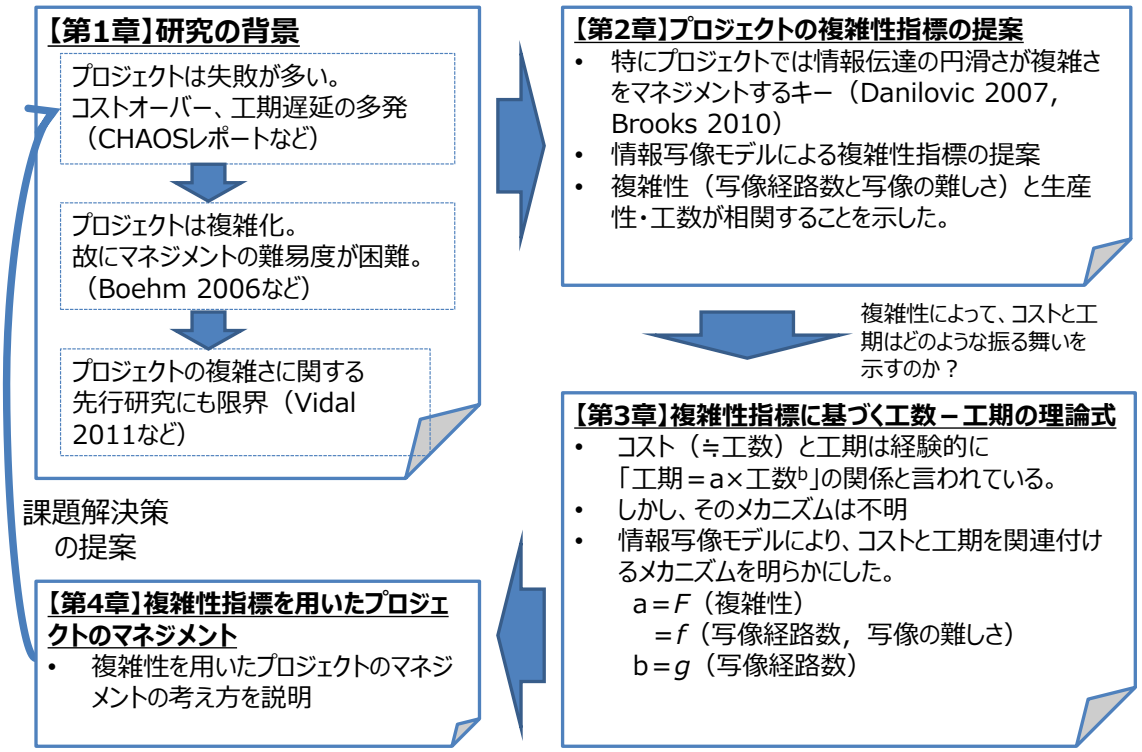


図 1-1　研究の概要（論文の構造）

Figure 1-1 Structure of This Research

【第２章】
プロジェクトの
複雑性指標の提案

２．　プロジェクトの複雑性指標の提案

2-1 ソフトウェア開発プロジェクトの現状

2-1-1 ソフトウェア開発プロジェクトにおける複雑性のマネジメントの現状

ソフトウェア開発プロジェクトは，設計情報を開発者間で伝達し，詳細化し，ソースコードを書くという，設計情報をやり取りするコストが開発費用の大半を占める．そのため，開発要員を増やしても，必ずしもプロジェクト全体の生産性が向上する訳ではない．逆に生産性が悪くなるケースも多い．何故ならば，開発要員の増加は開発要員間のコミュニケーションを複雑化させるためである[1]．このため，複雑なコミュニケーションが必要な作業内容を改善しなければ，効率的な開発（生産性向上や工数削減）は実現できない．

開発プロジェクトにおけるコミュニケーションに関する先行研究は，以下の 2 つの観点から分類することができる．1 つ目はコミュニケーションによる情報伝達の難しさの要因に関する研究である．2 つ目は適切な相手とのコミュニケーションの有無に関する研究である．まず，1 つ目の観点として，開発プロジェクトのコミュニケーションを改善するために，犬塚[2][3]はソフトウェア開発プロジェクトの開発者間の情報のやり取りを調査した．その結果，プロジェクト内でやり取りされる情報の質や量，媒体，送受信者の能力差によって情報伝達の難易度が変わることを示した．一方，2 つ目の観点として，Sosa ら[4]は，エンジン開発プロジェクトにおいて，部品間に相互依存があるならば，インタフェースを決めるためにその開発者間のコミュニケーションも必要となることに着眼した研究を行った．すなわち，開発者同士のコミュニケーションと同時に，エンジンを構成する部品間の相互依存を調査した．そして，部品間の相互依存と，各部品の開発者間で行われるコミュニケーションの有無を比較する方法を提案している．以上より，無駄なコミュニケーションや必須のコミュニケーションの欠如などプロジェクト内コミュニケーション状態を把握することが，品質や生産性の向上に有益であると述べている．これらの研究は，プロジェクト内のコミュニケーションについて以下の点を定性的に明らかにしている．すなわち，プロジェクト内の複雑なコミュニケーションでは，そもそものコミュニケーションの有無，及びそのときやり取りされる情報の質や量，伝える媒体，送受信者の能力差が，開発されるプロダクトの品質やプロジェクトの生産性・工数に影響を与えるということである．

しかし，このどちらの先行研究も，何故，開発作業では複雑なコミュニケーションが必要となるのか明らかにされていない．これは，他の先行研究でも同様である．ソフト

ウェア開発プロジェクトのプロセスに特化したものではないが, 前述の犬塚と同様, von Hippel[5]や Draft and Lengel[6]も人と人とのコミュニケーションについて研究を行っている．そして，その情報の質や量，媒体，送受信者の能力差に関連した情報伝達の難しさを指摘している．しかし，これらの研究では，コミュニケーションが複雑化する仕組みは明らかにされていない．一方，前述の Sosa らの研究と同様に，開発プロセスとプロダクトの関連性については Eppinger ら[7][8][9][10]や Danilovic and Browning[11][12]も取り組んでおり，マトリックス（Design Structure Matrix[13]）を用いた方法論を提案している．しかし，これらのプロセスとプロダクトの両面からの研究においても，コミュニケーションが複雑化する仕組みは明らかにされていない．実際，ソフトウェア開発では，良い開発プロセス（作り方）が良いプロダクト（製品）を生むことを前提に考えられているものの, プロダクトとプロセスの関連は十分に解明されていない[14]．すなわち，プロセスにおけるコミュニケーションが，プロダクトにどのように関連しているのか，そして，その関連性がコミュニケーションの複雑化にどのように寄与しているのか，明らかにされていない．更に，その関連性が，プロジェクトの生産性・工数にどのように影響を及ぼすのか，その仕組みも明らかにされていない．

また，上記の研究では，コミュニケーションの複雑さとプロジェクトの生産性や工数等の各種マネジメント指標との関係については明確に示されていない. 実際の開発プロジェクトにおいても White and Fortune[15]が示したようにプロジェクトで利用される主要なマネジメントツールはガントチャートによる作業の進捗管理ツールであり, 複雑性のマネジメントに関するツールは活用されていない.

このため，筆者は過去にプロジェクトの複雑性に関する研究を行ってきた．まず，先行研究では複雑性を相互依存性と難易度の積と定義し, プロジェクトを構成する要素間の複雑な関係を指標化した．その指標値の大小により，プロジェクトの複雑さの度合を知ることはできた．まず，複雑性の概念の初期案を提示し[16]，アニメ・CG 製作プロジェクトの複雑性を机上で検証した[17]．次に算出方法に改良を加え，プロジェクトでの活用方法を机上のモデルを用いて説明した[18]．また複数プロジェクト間で部品を流用する場合の効果について，その分析方法を提案した[19]．最後に実プロジェクトにおける設計課題数と複雑性の相関を検証した[20]．しかし，筆者の過去の研究における複雑性の定義ではプロジェクトにおける設計情報の写像経路数や各経路における写像の難しさを明らかにしていない．従って，設計情報のやり取りがコミュニケーションを複雑化させるか否かの説明には至っていない．当然，写像経路と生産性や開発工数の関連

は考慮していなかった．また，その複雑性と生産性の相関については言及しているものの，開発工数との相関については言及していない．しかも，生産性と複雑性の相関は机上モデルによる説明に留まり，実プロジェクトでの検証は行っていない．

なお，複雑性について Lindeman et al.[21]では工学の分野において標準的な定義は存在しないと述べている．同書では，複雑性を要素間の相互作用から生じるものという主張もある[22]．一方，同書では Ehrlenspiel[23]は技術システムにおいてはその構成要素の量とそれぞれの結合性に依存するが，モノの複雑性は変数の量，結合度，凝縮性，運動量に依存するという主張も紹介している．このように複雑性の定義は一意に定まったものではなく，個々でその定義を行っている．このため，従来は複雑性の定義が明確ではなかった．

以上のように，従来の研究では，コミュニケーションの難しさを生じる要因，コミュニケーションの有無が及ぼす影響について研究が行われ，コミュニケーションがプロジェクトの品質や生産性に影響を及ぼすことは定性的に指摘されていた．しかし，ソフトウェア開発プロジェクトのプロセスやプロダクトを構成する要素間の複雑な関係を明確且つシンプルに表した，実プロジェクトにおいて有効な複雑性の研究は行われていなかった．

2-1-2本章の目的と概要

本研究では，ソフトウェア開発プロジェクトの要素間の複雑な関係をシンプルに表すモデルを提示し，その有用性を示すことを目的とする．本モデルでは，設計情報の写像経路に着目したプロジェクトの複雑性指標を用いる．そして，その複雑性指標と生産性（1 人あたりの平均作業日数）及び開発工数が相関することを，実際の開発プロジェクトを対象に実証的に検証する．これにより，プロジェクトの生産性向上や工数削減を実現するためには設計情報を写像する経路の冗長性を低減することが必要であり，そのためには提示したモデルが有用であることを示す．

以下に，本研究の概要を述べる．まず，公理的設計論[24]における設計情報の写像経路を簡素にモデル化することがコミュニケーションの複雑さを緩和し，コスト低減に繋がると考えた．そこで 2-2 章では，公理的設計論を発展させたプロジェクトモデルとその複雑性指標を構築する．提案した複雑性指標は生産性及び開発コストと相関する指標であることは勿論，以下の観点で先行研究の課題を解決している．まず第 1 に，プロジェクトを構成する要素間の相互依存性のみならず，設計情報の写像の難しさも考慮した

複雑性指標を導出する．そして，開発するプロセスだけでなく開発するプロダクトとの関連をその写像経路で示すことで先行研究の課題に取組む．2-3 章では，複雑性指標の具体的な算出方法を紹介する．2-2 章で紹介したモデルを行列で表現する方法を示す．その行列計算によって，プロジェクトで形成される写像経路の本数が算出できることを示す．また，各経路の写像の難しさの算出方法も同時に示す．さらに，2-4 章において，実際にゲームソフトウェア開発を行った 21 の開発プロジェクトから複雑性と生産性，開発工数の相関を実証的に検証する．その結果，提案した複雑性指標と生産性及び開発工数に強い相関関係があることを示す．2-5 章では，検証結果を考察し，開発工数や生産性（1 人あたりの平均作業日数）の大半が写像経路の冗長性に起因していることを説明する．

なお，生産性とは，本来，生産物の量を投入物の量で割った値として表される．本研究では，更に生産性を 2 つの要素に分解し，①1 人あたりの平均作業日数（全工数／開発要員数）と②1 人日あたりの成果物量（生産物の量／全工数）の積で表されると考えた．しかし，本研究における生産物のひとつである音声データは，一般的にソフトウェア開発で用いられる生産物の量であるソースコード行数と同じ単位で測ることが出来ず，プロジェクトの全ての生産物の量をひとつの数値で規定することが出来なかった．従って，②を規定出来なかった．そのため，以降では，生産性を狭義に捉え，①1 人あたりの平均作業日数という作業効率を表す指標のみを生産性として定義する．

2-2 ソフトウェア開発プロジェクトのモデル化とその複雑性指標の構築

まず，本研究ではソフトウェア開発プロジェクトをどのように捉えるのか，その枠組みとなるプロジェクトのモデルについて説明する．

2-2-1 公理的設計論の定義と本研究における拡張点

公理的設計論とは，独立公理（具備する機能は独立性を高く設計した方が良い設計である．）と情報公理（設計に必要な情報量は少ない方が良い設計である．）の 2 つの公理から成る，良い設計を行うための設計方法論である[24]．その理論の中で，設計とは顧客領域，機能領域，実体領域，プロセス領域の 4 つの領域間における設計情報の写像過程と述べられている．顧客領域とは顧客が期待する商品等，顧客のニーズを扱う領域である．また機能領域は顧客の要求仕様が要求機能と制約条件で示された領域である．実体領域は設計解を示す領域である．そしてプロセス領域は生産条件を示す領域である．設計課題として顧客領域があり，その設計解として機能領域が規定される．また，設計課題としての機能領域は実体領域が設計解となる．設計課題としての実体領域はプロセス領域がその設計解となっている．このようにそれぞれは設計課題と設計解の関係を成している．

本理論では，機械工学の分野のみならず，ソフトウェア自体やビジネスの構造等も利用可能な設計方法を説明している．以上のように一般論としての設計理論であるため，上記のように各領域の意味付けはされているものの，具体的な要素は規定されていない．一方，本研究は，ソフトウェア開発プロジェクトを研究対象としている．そこで，次節では，公理的設計論に基づいたソフトウェア開発プロジェクトを構成する構成要素を規定し，写像関係をモデル化する．

2-2-2 ソフトウェア開発プロジェクトのモデル化

プロセス領域に開発者（Team），作業（Activity），成果物（Artifact），実体領域に機能（Function），部品（Component），機能領域に要求仕様（Requirement），顧客領域にニーズ（Needs），基本要件（Feature）を要素として抽出し，それらの要素間で写像が行われる経路をモデル化した．前節で説明した通り，この要素間の関係も，設計課題と設計解の関係となっている．設計課題としてニーズ（Needs）があり，その設計解が基本要件（Feature）である．基本要件（Feature）を設計課題として捉えれば，要求仕様（Requirement）が設計解である．このような関係が開発者（Team），作業

（Activity）まで続いている．

例えば，実証検証を行ったゲーム開発プロジェクトでは，要求仕様（Requirement）のひとつとして「ゲームのストーリーが面白いこと」という項目があった．すなわち，設計課題として，ゲームを面白くするためにどうすべきか，課題が設定されたのである．更にその設計解として，ゲーム開発プロジェクトでは，ポイント計算機能（Function）を付加することとした．すなわち設計解としてポイント制のゲームにすることとした．次に，設計課題として，ポイント計算機能の実現という課題が設定されたので，設計解として，メインフレームとアダプタ（Component）を使うこととした．メインフレームとアダプタで機能を実現するためには（設計課題），ソースコードという成果物（Artifact）が必要である（設計解）．ソースコードを書くためには（設計課題），プログラミングという作業（Activity）を行うことが必要（設計解）である．プログラミングを行うには（設計課題），そのスキルを持ったチームプログラマやプログラマーを確保し，開発体制（Team）を整えることが必要（設計解）である．このように設計情報が設計課題から設計解という形に変異しながら，プロジェクト内を流通していく過程として，開発プロジェクトを捉え，それをモデル化した．次に，これらの関係を更に整理し，汎用的なモデルとして説明を行う．

図 2-1 に示した丸印はプロジェクトを構成する個々の要素である．例えば，プロセス領域の Team においては，アーキテクトやプログラマーといった開発者の役割を個々の丸印で表現している．また，図中に示した実線は個々の要素間の関連を示している．実線で結ばれた要素間で設計情報が写像されることを表している．

このように設計情報が，設計課題と設計解という形に変化しながら流通するのがプロジェクトである．次に，各要素の定義やその関連性について，更に詳しく述べる．

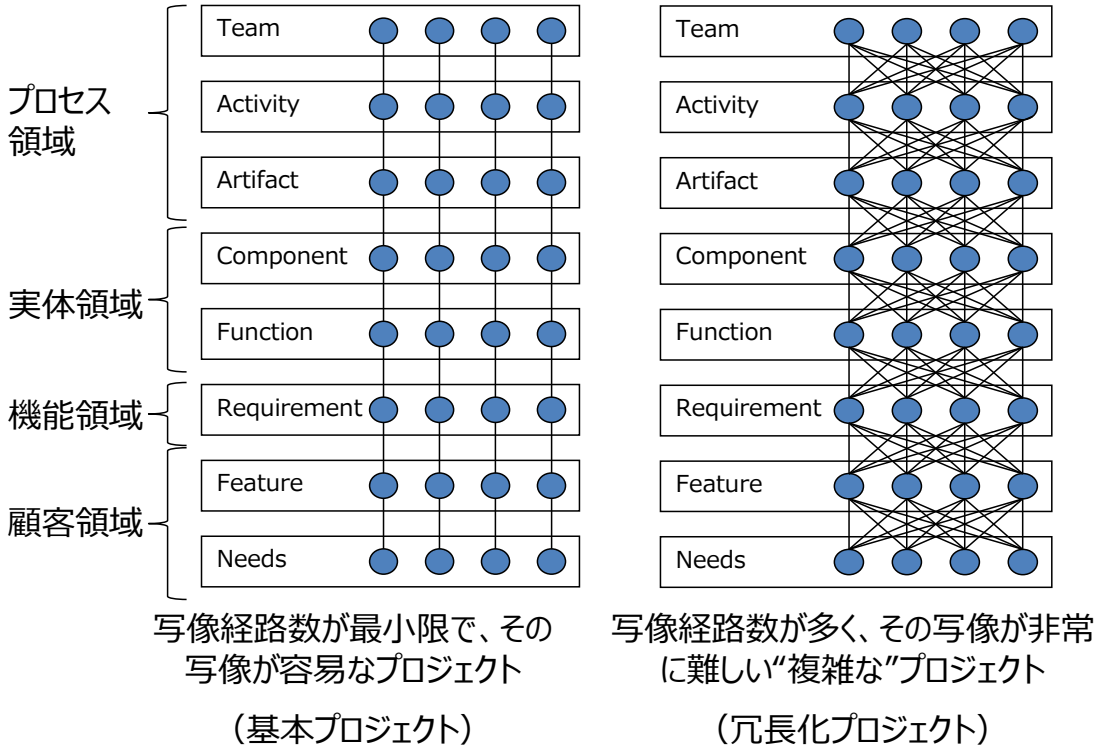


図 2-1 ソフトウェア開発プロジェクトの情報写像モデル

Figure 2-1 Information Mapping Model of Software Development Project

以下で各要素の関連について説明を行う．Team とはプロジェクト内で分担される様々な役割毎に配置されるチーム又は個人を指す．例えば，前述の例の通りアーキテクトやプログラマー等が該当する．Activity は Team が実施する仕事であり，Artifact を生み出す．例えば仕様書の作成等が該当する．Artifact は Activity により生産されるものであり，例えば設計書等を指す．以上をまとめると Team が Activity を行うことで Artifact を生み出すという関係にある．従って，プロセス領域内の要素の関連は，図に示したように Team と Activity を実線で結び，Activity と Artifact を実線で結んだ．Team は何かしらの Activity を行うことで Artifact を参照または更新するのであって，Activity を介さずに Artifact と関連することは無いと考えた．次に，Component は Artifact を実装したものであり，Function を有する．Function は Component の持つ役割を示し，これにより Requirement を実現する．従って，実体領域の要素の関連はプロセス領域の Artifact と Component を実線で結び, Component と Function を実線

で結び,更に次で説明する機能領域にある Requirement と Function を実線で結んだ．あくまでも Component が保有する Function が Requirement を実現するので，Component が Requirement を実現する訳では無い．そのため Component と Requirement との関連付は無いと考えた．また同様に Function は Component に具備されてはじめて有効となるものなので，Component を介さずに Function と Artifact が直接関連することは無いと考えた．機能領域にある Requirement は，システムとして具備すべき Function を規定したものである．例えば“システムにより△△を行い□□を出力すること.”のように記述される.Requirement は次に説明するように Feature を具体化したものなので Feature と実線で結んでいる[25]．更に Feature は，システムで実現すべき事項を規定しており，例えば“システムは××を実現すること.”のように記述される．Feature をより具体化したものが Requirement である．Needs は，例えば“我々は○○が欲しい.”というように顧客が欲しているものを示す．以上から，顧客領域では Feature と Needs を実線で結んだ[25]．また，その具体化の度合から Needs は Feature を介してのみ Requirement と関連を持つ[25]と考えた.

2-2-3 プロジェクトモデル内の写像過程と生産性及び開発工数

本節では，図 2-1 に示したモデルを用いてプロジェクト内の写像過程を説明する．前述の通り，先行研究では，①コミュニケーション時にやり取りされる情報の質や量，伝える媒体，送受信者の能力差，及び②そもそものコミュニケーションの有無が，生産性や開発工数に影響を及ぼすことが個々に明らかにされている．これら①②で述べられているのは人同士の情報伝達を前提としているが，この考え方は，本研究における上記の要素間の写像においても以降で述べるように応用が可能と考えた．その結果，プロジェクトの写像過程では,個々の要素の持つ写像の難しさと写像経路の冗長性の 2 点を個別にではなく，合わせて考慮することが，生産性や開発工数に影響を捉える上で必要であることが明らかになった.

2-2-3-1 要素の持つ写像の難しさと生産性及び開発工数

まず，上記①について，本研究で用いた各要素が持っている写像の難しさを生じる特性として応用し，表 2-1 にまとめた.

表 2-1　プロジェクトを構成する要素と写像における特性

Table 2-1 Project Model Element and Feature for Mapping

領域	要素	写像の難しさを決める特性
プロセス領域	開発者 （Team）	“開発者”のスキル、経験は“作業”の効率性に影響する。スキルが低く、経験が少ない程、写像が難しい。
	作業 （Activity）	“作業”の量や難しさは“成果物“の品質に影響する。量が多く、難しい程、写像が難しい。
	成果物 （Artifact）	成果物の量や品質は部品の質や量に影響する。量が多く、質が悪い程、写像が難しい。
実体領域	部品 （Component）	部品の質や量は保有する機能の質や量に影響する。質が悪く、量が多い程、写像が難しい。
	機能 （Function）	機能の質や量は要求仕様の充足性に影響する。質が悪く、量が多い程、写像が難しい。
機能領域	要求仕様 （Requirement）	要求仕様の項目数や曖昧さやは基本要件の充足性に影響する。曖昧で項目が多い程、写像が難しい。
顧客領域	基本要件 （Feature）	基本要件の曖昧さや要件数はニーズの充足性に影響する。曖昧で量が多い程、写像が難しい。
	ニーズ （Needs）	ニーズの曖昧さやその数は、基本要件の曖昧さを生じさせる。曖昧で数が多いほど写像が難しい。

このようにプロジェクトを構成する要素は，人同士のコミュニケーションでなくとも，やり取りされる情報の質や量，能力によって，写像を難しくする特性を持っていると考えた．

以降で各要素の持つ特性が，どのように写像を難しくするのか述べる．例えば，開発者(Team)のスキルや経験は，作業(Activity)に影響する．何故ならば，スキルや経験が無ければ，そもそも作業(Activity)として行うべき一つ一つの動作すら，正確に行うことが出来ないからである．また，作業(Activity)自体の量や質によっても，開発者(Team)のスキルや経験を活かした作業(Activity)を行えるかどうかが決まってしまう．当然，正確な動作が出来なければ正確に動作が出来るまでやり直し（写像のやり直し）が生じるため，生産性が低下し，それに伴った開発工数の増大が起きる．同様に，作業(Activity)の中で行う動作のひとつひとつの質は，当然成果物(Artifact)の質に影響を及ぼす．もし作業(Activity)量が多ければ，ひとつひとつの動作の質も低下することが考えられ，

それも同様に成果物(Artifact)の質に影響を及ぼす．質が悪ければ，何度も質が良くなるまで写像を繰り返すことになり，これも生産性の低下を招き，それに伴って開発工数が増加する．成果物(Artifact)の量が多い場合や出来上がりに高い質を求められている場合など，作業(Activity)にも高い正確性を求められるため，同様に写像を繰り返すことが想定される. その結果, 生産性や開発工数に影響することは言うまでもない. また，成果物(Artifact)と部品(Component)の関係についても同様である．成果物(Artifact)の質や量は部品(Component)に影響を及ぼし，そのため写像過程の手戻りを生む．部品(Component)と機能(Function)，機能(Function)と要求仕様(Requirement)，要求仕様(Requirement)と基本要件(Feature)，基本要件(Feature)とニーズ(Needs)についても同様である．

２-２-３-２写像経路の冗長性と生産性及び開発工数

前節では，プロジェクトを構成する要素の特性による写像の難しさと，それに伴って生産性や開発工数に影響が生じることを述べた．しかし，前述の②に関してはまだ述べていない．②では，開発されるプロダクトの部品構造を基準として，コミュニケーションをすべき開発者を規定していた．そして，その開発者同士が情報のやり取りをしているか否か, または本来ならばやり取りする必要のない開発者同士が情報のやり取りをしているか否か, というコミュニケーションの冗長性が生産性や開発工数に影響を及ぼすと述べている．本研究においては，写像経路は冗長な経路なのか否か,つまり必要な要素間のみを写像する経路になっているのか，という経路の冗長性の観点で②を応用することを考えた．

以下に写像経路の冗長性の影響を述べる．例えば，写像経路がある要素で交じりあうことがなく，つまり写像経路に冗長性が全く無く，Team から Needs まで写像するだけならば，前節で述べたように，その写像の正確性は表 2-1 で示した各要素の特性だけで決まる．生産性や開発工数も同様である．図 2-1 の左図に示したように写像経路が途中で分岐すること無く, Team のあるひとつの要素を示す丸印から Needs のひとつの要素を示す丸印まで，1 本の写像経路で表される場合である．しかし，図 2-1 の右図に示したように写像経路が途中で分岐し，Team のあるひとつの要素を示す丸印から Activity の全要素の丸印と実線で結ばれ，同様に Needs の要素まで複数の関連付けがなされている場合は, 個々の要素の特性だけでは写像の正確性は決まらない. 何故なら, 例えば，Artifact のあるひとつの丸印ともうひとつ別の丸印，それらと Component のひとつの

丸印に写像関係が存在している場合，（つまり，2 つの写像経路が存在している場合），Artifact から Component へ，もし前節で述べた特性により一方の経路で写像が正確に行われなければ，Component の要素は必要な設計情報を欠いてしまうこととなるからである．その結果，冗長な写像経路が増えるほど，生産性が低下し，それに伴い開発工数の増大が見込まれる．以上より，本研究で示した要素間の写像経路の冗長性は生産性や開発工数に影響を及ぼすと言える．

上記を踏まえ，本研究では，写像経路の冗長性と，前節で述べた個々の要素の特性も，正確な写像を行う上で同時に考慮すべきであると考えた．従って，以降では，これらの 2 つの観点からプロジェクトの複雑性を導出していく．

2-2-4 ソフトウェア開発プロジェクトの複雑性指標の導出

プロジェクトを構成する写像経路と，その写像経路の写像の難しさを用い，プロジェクトの生産性と相関する複雑性指標を定義する．前節で述べたように，要素間の関係が 1 対 1 ならばその写像経路の写像の難しさが生産性と相関する．また，写像経路の冗長さが増すほど生産性が低下する．よって，プロジェクトの複雑性を以下の式(2-1)で定義する．そして，写像経路の複雑性とプロジェクトの開発作業の生産性（開発要員 1 人あたりの平均作業日数）の関係を式(2-2)により定義する．

なお，各変数は以下のように定義する．

α; 複雑性
β; 基本プロジェクトの写像の難しさ
γ; 冗長化したプロジェクトの写像の難しさ
i; 開発要員数
k; 1 本平均の写像の難しさ
l; 全経路数
m; 並行経路数
n; 冗長な経路数
C; 全開発工数
D; 1 人あたりの平均作業日数
W; 複雑性指標の総和
以降では,上記変数を用いる．

$$\alpha = k \times m \times \left(1 + \frac{n}{m}\right) \tag{2-1}$$

$$\alpha \propto D \tag{2-2}$$

並行経路数(m)とは図 2-2 における実線で示した経路の数である．図 2-1 の左図と同様，列を跨らない場合を意味する．冗長な経路数(n)とは図 2-2 における点線で示した経路の数である．すなわち，列を跨った写像経路数を意味する．

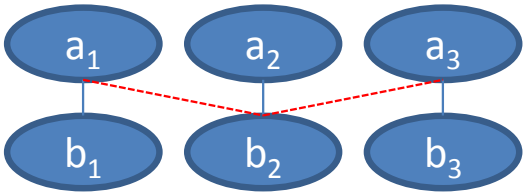


図 2-2　並行経路数と冗長な経路数の定義

Figure 2-2 Number of Parallel and Redundant Path

2-2-5複雑性の定義式の変形

複雑性を式(2-1)で定義したが，実際にプロジェクトの複雑性を求めることは難しい．何故ならば，そもそも経路 1 本平均の写像の難しさがわからないからである．しかし，式(2-1)を変形すると“個々の経路の持つ写像の難しさの平均値(k)×全経路数(l)”となる．言い換えれば，複雑性とは，個々の経路の写像の難しさの和となる．以下にそれを示す．

$$\alpha = k \times m \times \left(1 + \frac{n}{m}\right)$$

$$= k \times m \times \left(\frac{m+n}{m}\right)$$

$$= k \times m \times \frac{l}{m}$$

$$= k \times l$$

$$= \sum_{1}^{全経路数} (\textit{冗長化プロジェクトの各経路の写像の難しさ}) = \gamma \qquad (2\text{-}3)$$

2-2-6複雑性からの逆算方法（基本プロジェクトの求め方）

複雑性は，冗長化したプロジェクトにおける写像の難しさ(γ)に等しい．従って，冗長化したプロジェクトの写像の難しさが分かれば，冗長化していないプロジェクトの写像の難しさも逆算できる．冗長化していないプロジェクトの写像の難しさを算出できれば，どの程度冗長化によってプロジェクトの複雑性が増したのか測定することができる．まず，元のプロジェクトの写像の難しさを β とすると式(2-3)は次式となる．

$$\alpha = k \times m \times \left(1 + \frac{n}{m}\right) = \beta \times \frac{l}{m} = \gamma \qquad (2\text{-}4)$$

式(2-4)における基本プロジェクト(β)は，図 2-3 のように，冗長な経路が存在しない状態と定義する．

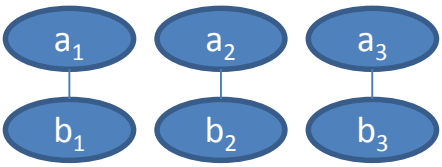


図 2-3　基本プロジェクトの定義

Figure 2-3 Basic Project Model

従って，式(2-4)より，基本プロジェクトは以下のように算出できる．

$$\beta = \gamma \times \frac{m}{l} \tag{2-5}$$

$\alpha(=\gamma)$と β を比較することで，プロジェクトの冗長化による複雑性の増分を測ることができる．

2-2-7 複雑性と開発工数の関係式

本節では開発工数と複雑性の関係式を導出する．式(2-2)より，複雑性は，1 人あたりの平均作業日数と相関すると考えられている．従って，開発工数（1 人あたりの平均作業日数と要員数の積）は，複雑性に要員数を掛け合わせたものと相関すると考えられる．その時，プロジェクト全体の複雑性は，プロジェクトに存在する全経路で，全ての要員が携わって写像していると考え，算出されている．そして，その複雑性と 1 人あたりの平均作業日数が相関するとしている．しかし，実際のプロジェクトでは，開発中に必ずしも全経路の写像に全ての要員が携わっているとは限らない．従って，開発工数を算出する上では，開発期間中のある時点において，写像されている経路数に対して，その写像に携わっている開発要員数を考慮した方が精度を高く開発工数を求めることができると考えた．そこで，開発期間中のある時点での写像経路数に対する複雑性と，それに携わる開発要員数を元に，開発工数を積み上げ，全開発工数を算出する方法を以下に述べる．

まず，開発期間中のある時点 t で，冗長な写像経路数が j 本のときの複雑性，1 人あたりの平均作業日数，開発要員数，開発工数等を

α_j；j 本の時の複雑性

C_j；j 本時の開発工数（人日）

D_j；j 本時の 1 人あたりの平均作業日数（日）

i_j；j 本時の開発要員数（人）

W；複雑性指標の総和

のように定義すると，式(2-2)より，t 時点の j 本の写像経路で行う開発工数は

$$C_j = D_j \times i_j \propto \alpha_j \times i_j \tag{2-6}$$

と表される．上記式(2-6)に対して，開発期間中，プロジェクトとして取り得る経路数は 0 から n 本までであるので， その総和は次式となる．

$$C = \sum_0^n (D_j \times i_j) \propto \sum_0^n (a_j \times i_j) \tag{2-7}$$

すなわち，全開発工数(C)と複雑性(α)は式(2-7)として表されると考えた．

また，式(2-7)は，開発期間中，開発要員数が変化しない場合は，i_jが一定値 i となるので，以下のように表される．

$$C = \sum_0^n D_j \times i \propto \sum_0^n \alpha_j \times i = k \times m \times i \times \left(n + \frac{n^2}{2m}\right) = W \times i \tag{2-8}$$

開発期間中，本研究では開発要員数(i)は一定であったので，全開発期間を通した開発工数との関係を式(2-8)のように設定した．これらを踏まえると式(2-8)は，

$$C \propto W \times i \tag{2-8'}$$

のように表される．

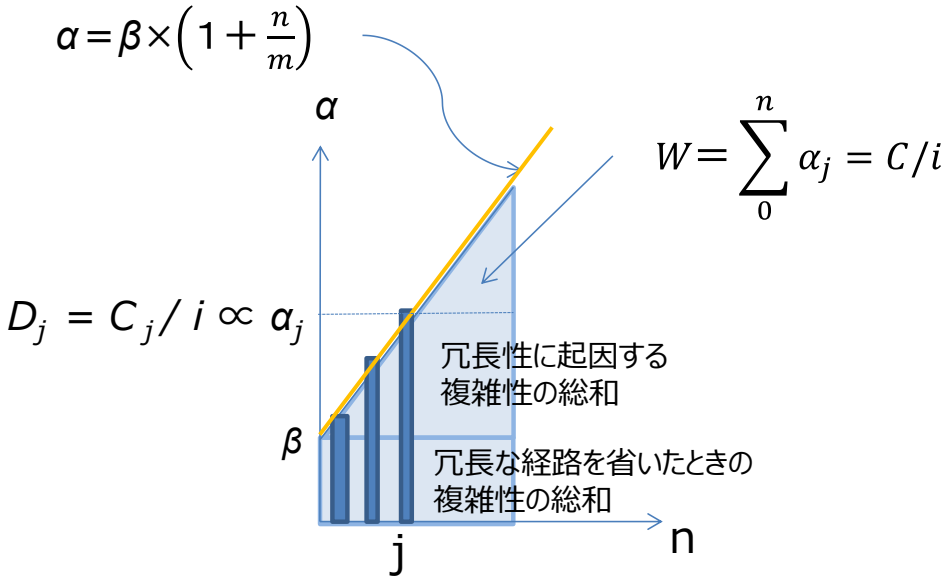


図 2-4　複雑性指標と開発工数

Figure 2-4 Project Complexity and Man-hour

図 2-4 は複雑性(α)と冗長な経路数(n)の関係式と x 軸，y 軸で作られる台形を示している．複雑性指標の総和(W)とはこの台形の面積を指す．式(2-8')より，プロジェクトの全開発工数(C)は，この面積と要員数(i)の積である．そして，台形の中に描かれている縦長の長方形は，冗長な経路数 j 本の時の複雑性を示す．また，式(2-1)の y 切片は図中の基本プロジェクトの写像の難しさ(β)を示しているので，冗長性が無いときの生産性に等しい．すなわち，冗長性が無い場合の複雑性指標の総和は，y 切片を縦，経路数を横とする長方形の面積に等しい．次に，冗長性に起因する生産性（1 人あたりの平均作業日数）は冗長なプロジェクトの生産性（冗長な経路数 n 本の時の y 軸の値）から基本プロジェクトの生産性（y 切片の値）を差し引いた値となる．それは図 2-4 の上三角形の高さで示される．すなわち，冗長性に起因する複雑性指標の総和は上三角形で示される面積に等しい．複雑性指標の総和と開発要員数の積はプロジェクトの全開発工数である．従って，冗長性に起因する開発工数は図の上三角形の面積に要員数を掛け合わせることによって求めることが出来る．同様に冗長性が無い場合の開発工数は長方形の面積と要員数の積で求まる．

2-3 写像経路本数と写像の難しさの算出方法

プロジェクトのモデルとその複雑性の算出方法は前章で述べた．しかし，その算出に用いる写像経路数や写像の難しさの計算方法は，まだ述べられていない．そこで本章では，どのように写像経路数や写像の難しさを算出するのか説明する．

2-3-1 行列を用いた算出の基本原理

行列を用いることで，経路数，写像の難しさの算出が可能である．その算出方法を次に示す簡易モデルで説明する．

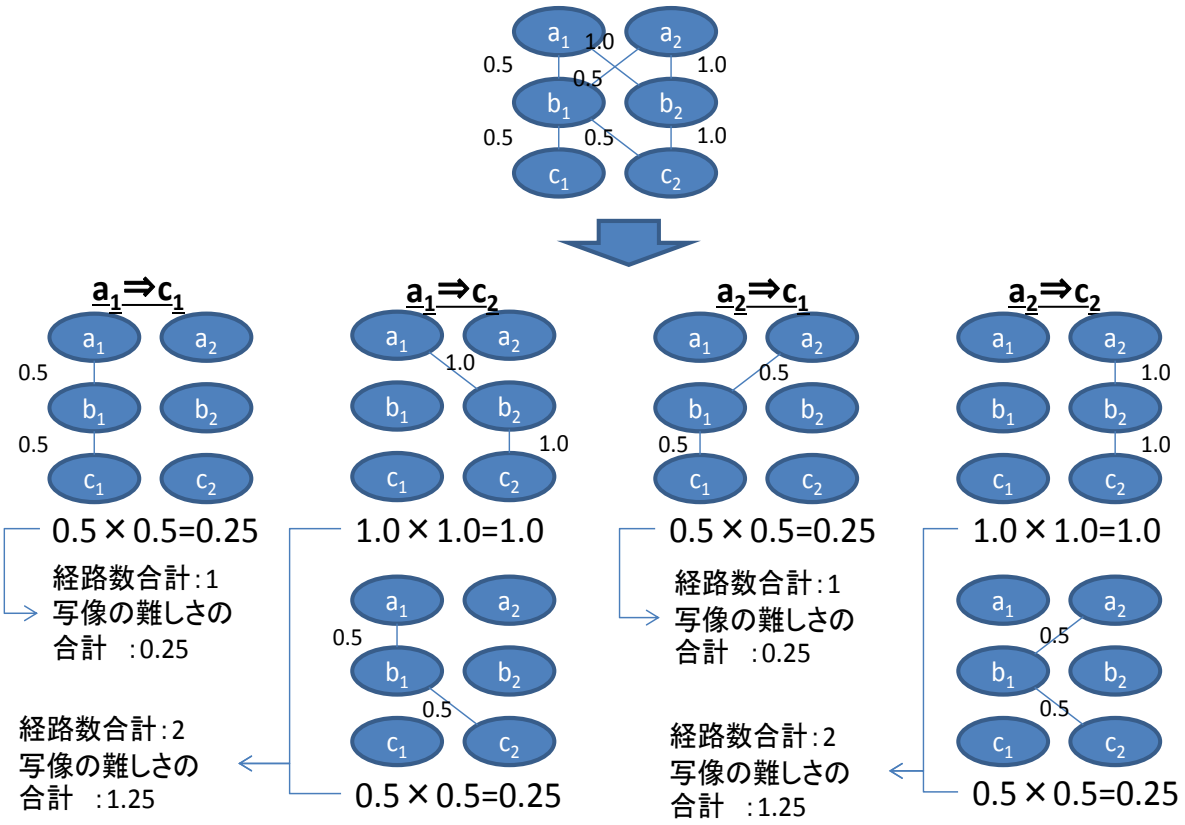


図 2-5　写像経路数と各経路の写像の難しさの算出方法

Figure 2-5 Numbers of Mapping Paths and Difficulty of Mapping Paths

図 2-5 の上図で示したモデルの丸印(a_1~c_2)はプロジェクトを構成する要素を示す．丸印同士を結んだ実線は写像関係の有無を示す．実線上に付与した数値はその写像経路における写像の難しさを示している．このモデルに描かれている写像経路を全て抽出したものが図 2-5 の下図である．a_1 から c_1 への写像は 1 本のみ，a_1 から c_2 への写像経路は 2 本，a_2 から c_1 への写像経路は 1 本，a_2 から c_2 への写像経路は 2 本，それぞれ具体的な写像経路と共に示している．また，各モデルの写像の難しさの算出式が各モデルの下に示してある．各写像経路の写像の難しさの積がその算出式の解となっている．また，その積の合計値と前述の写像経路数を矢印の先に，合わせて記載した．

しかし，このように，図から個々の写像経路を抽出した上で写像経路数や写像の難しさを算出することは，モデルが複雑化するほど困難な作業となってしまう．そこで，行列を用いて算出する方法を考えた．まず写像経路数の算出方法は次式(式 (2-9))となる．

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \left\{ \overset{\begin{matrix} b_1 & b_2 \end{matrix}}{\begin{matrix} c_1 \\ c_2 \end{matrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}} \times \overset{\begin{matrix} a_1 & a_2 \end{matrix}}{\begin{matrix} b_1 \\ b_2 \end{matrix}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}} \right\} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$= \overset{\begin{matrix} a_1 & a_2 \end{matrix}}{\begin{matrix} c_1 \\ c_2 \end{matrix}\begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \qquad (2\text{-}9)$$

すなわち，図 2-5 で示した各要素(a_1~a_3,b_1~b_3)間に関連が有る場合には行列の成分値を 1 に，関連性が無い場合には行列の成分値を 0 に設定し，行列の積を取ることとする．積として得られた行列の成分値は，要素間を結ぶ写像経路数を示す．a_1 は b_1,b_2 と，a_2 は b_1,b_2 と関連がある．従って，式(2-9)で要素 a と b の関連を示した行列の成分値は全て 1 を設定した．同様な考え方から b と c の関連も行列の成分値として示した．その結果得られた行列の解では，a_1 と c_1 の成分値は 1，a_1 と c_2 は 2，a_2 と c_1 は 1，a_2 と c_2 は 2 となっている．これは図 2-5 で示した写像経路数とも一致する．

次に，写像の難しさの算出式を示す(式(2-10))．

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \left\{ \begin{matrix} c_1 \\ c_2 \end{matrix} \overset{b_1}{\underset{}{\begin{pmatrix} 0.5 & 0 \\ 0.5 & 1 \end{pmatrix}}} \times \begin{matrix} b_1 \\ b_2 \end{matrix} \overset{a_1}{\underset{}{\begin{pmatrix} 0.5 & 0.5 \\ 1 & 1 \end{pmatrix}}} \right\} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$= \begin{matrix} c_1 \\ c_2 \end{matrix} \overset{a_1}{\underset{}{\begin{pmatrix} 0.25 & 0.25 \\ 1.25 & 1.25 \end{pmatrix}}} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \qquad (2\text{-}10)$$

すなわち，行列の成分値に 2 つの要素間の写像の難しさを設定すれば，行列の積で得られる成分値は複数要素間を経由した写像経路の難しさを示す．つまり，上記と同様に a_1 と b_1 写像の難しさは 0.5，a_1 と b_2 とは 1.0，a_2 と b_1 は 0.5，a_2 と b_2 は 1.0 を行列の成分値として設定した．要素 b と c についても同様である．その結果得られた行列の解では，a_1 と c_1 の成分値は 0.25，a_1 と c_2 は 1.25，a_2 と c_1 は 0.25，a_2 と c_2 は 1.25 となっている．これは図 2-5 で示した写像の難しさとも一致する．

2-3-2 冗長な写像経路数の算出方法

冗長な写像経路数を求める方法について以下に述べる．前節で求めた写像経路数を示す行列の積と単位行列の差が冗長な写像経路数を表している．単位行列は冗長性のない状態であるので，得られた行列から引き，その行列の成分値の総和を取れば冗長経路数の合計値となる．例えば図 2-5 では，次式(式(2-11))となる．

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \left\{ \begin{matrix} c_1 \\ c_2 \end{matrix} \overset{b_1}{\underset{}{\begin{pmatrix} 1 & 0 \\ 2 & 2 \end{pmatrix}}} \times \begin{matrix} b_1 \\ b_2 \end{matrix} \overset{a_1}{\underset{}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}} \right\} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$= \begin{matrix} c_1 \\ c_2 \end{matrix} \overset{a_1}{\underset{}{\begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}}} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \qquad (2\text{-}11)$$

すなわち，第 1 項には図 2-5 における冗長経路数として式(2-9)の解を設定し，第 2 項には単位行列を設定している．この単位行列とは，冗長性が無い状態，つまり，a_1 から b_1，そして c_1 へ写像される経路，及び a_2 から b_2，そして c_2 へ写像される経路を示している．そして，式(2-11)より得られた解の成分値の合計値から，図 2-5 で示したモデルの冗長経路数は 4 本となる．

2-3-3写像経路の難しさの設定方法

前節までは写像経路の本数とその経路の写像の難しさの算出方法を述べた．しかし，その写像の難しさをそもそもどのように数値化するのか，その算出方法はまだ述べられていない．従って，本節では，個々の要素の写像の難しさから要素間の写像経路の難しさの算出方法を説明する．

まず，前述(表 2-1)の特性を考慮し，1 を基準として各要素の持つ写像の難しさを評価し定量化する．基準値 1 に対して，値の幅を等間隔とするため，各要素の定量化にあたっては 0＜写像の難しさ＜2 とする．但し，要素間に相互依存が無い場合は 0 である．また，1 以上は写像が難しく，1 以下は写像が容易であるものとする．そして，各要素の持つ難しさの相乗平均をその要素間の写像経路に対する写像の難しさとする．

相加平均ではなく，相乗平均とした理由について述べる．例えば，1 を基準としたとき，要素 A の難しさ＝0.5 であるならば，それは基準値 1 の 0.5 倍の難しさであることを意味する．同様に要素 B の難しさ＝1.7 であるならば，それは基準値の 1.7 倍の難しさであることを意味する．すなわち，要素に付与される数値は，基準値の何倍の難しさであるかを示す係数である．従って，本研究では相乗平均を取ることとした．なお，例えば，0.01 と 1.9 という値の平均を求めると，相加平均は 0.955（=（0.01＋1.9）÷2）となり，相乗平均は 0.138（=$\sqrt{(0.01 \times 1.9)}$）となる．このように相加平均では，1.9 という大きい値に引きずられた平均値となり，この 2 つ平均値は必ずしも一致するものではない．相乗平均は対数をとったものの平均であり，基準値の倍数のような係数の平均に用いられるものである．他の例として，銀行の預金利率の平均利回りなどのような係数も，上記と同様に相加平均ではなく，相乗平均で求められる．

以上により，相乗平均を取った値は，式(2-10)で示したように，各行列成分値として各行列に設定され，プロジェクトの写像の難しさを求めることに利用される．各要素の持つ写像の難しさの相乗平均値を成分値とした行列の積を行列 A とし，プロジェクトの写像の難しさを式(2-12)で定義する（a_{ij} は行列 A の成分値）．

冗長化したプロジェクトの写像の難しさ＝|A|

$$= \Sigma\, a_{ij} \tag{2-12}$$

2-3-4写像の難しさと冗長性の算出方法の補足とまとめ

前節までは，簡易モデルとして示した図 2-5 を基に式(2-9)(2-10)を示した．しかし，実際の開発プロジェクトでは図 2-1 に示したようなモデルとなるため，本節では実際の

開発プロジェクトにおける写像経路数及び写像の難しさの算出方法を説明する．その際，図 2-1 に示した基本パターンの算出方法では対処できない 3 つの派生パターンがある．本節では，その 3 つの派生パターンの算出方法も説明する．

２-３-４-１基本パターン

実際の開発プロジェクトでは，式(2-9)(2-10)に該当するものとして式(2-13)を用いる．

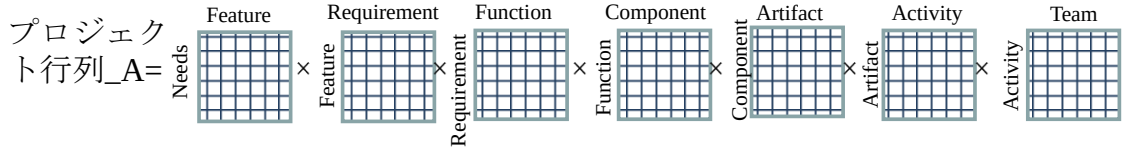
$$\begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix} \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix} \begin{pmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1} & e_{n2} & \cdots & e_{nn} \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{pmatrix} \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nn} \end{pmatrix} \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n1} & h_{n2} & \cdots & h_{nn} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}$$

Needs vector / Feature-Needs matrix / Requirement-feature matrix / Function-Requirement matrix / Component-Function matrix / Artifact-Component matrix / Activity-Artifact matrix / Team-Activity matrix / Team vector

プロジェクト行列_A

(2-13)

以降では，本式（プロジェクト行列_A）を，模式的に表現し，同じく式(2-13)として下記のように記載する．

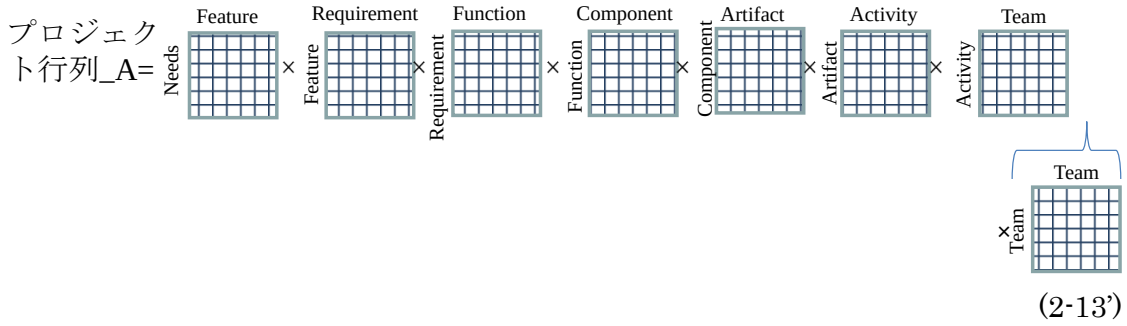


(2-13)

プロジェクト行列 A は，すなわち，図 2-1 で示したプロジェクトの基本パターンとなる各要素を行または列として作成したマトリックスで構成されている．左から，Needs と Feature 間での写像を第 1 項で示し，次に Feature と Requirement の関連を行列で表現した．同様に Requirement と Function，Function と Component，Component と Artifact，Artifact と Activity，そして最後に Activity と Team の写像関係を行列で示した．そのマトリックスに前述の通り 1 又は 0 を設定し，これら 7 つの行列の積をとれば，写像経路数を求めることができる．また同様に各要素の写像の難しさの相乗平均値を設定し，これら 7 つの行列の積をとれば，該当の写像経路の難しさを求めることができる．

２-３-４-２派生パターン 1

次に，派生パターンの 1 つ目を説明する．式(2-13)では同一カテゴリ同士(例えば Team 内の要素間)の写像を表現できないため，数式の変更が必要となる．その場合は，行と列を同一カテゴリとしてその関係性を行列化し，式(2-13)に挿入して同様な計算を行えば良い(式(2-13’))．



(2-13’)

２-３-４-３派生パターン 2

派生パターンの 2 つ目として，図 2-6 に示したように最下層の要素まで写像経路が続いていないモデルも式(2-13)では表現し切れないため，数式の変更が必要となる．このようなモデルでは，変更しないまま式(2-13)を用いると写像経路数も写像の難しさも誤った値が算出されてしまう．

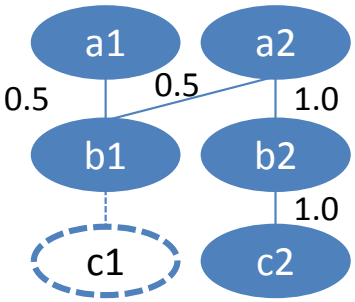


図 2-6（例）最下層の要素まで写像経路が無いモデル

Figure 2-6 Sample Model which Lacks a Mapping Path to the Bottom Layer

具体的には，この例では本来 a_1 から b_1 を経由し，再度 a_2 に戻って b_2，c_2 と辿る写像経路を示したい．しかし，図中に示した値を各写像経路における写像の難しさとすると，式(2-14)のようになる．

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \left\{ \begin{matrix} & b_1 & b_2 \\ c_1 \\ c_2 \end{matrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{matrix} & a_1 & a_2 \\ b_1 \\ b_2 \end{matrix} \begin{pmatrix} 0.5 & 0.5 \\ 0 & 1 \end{pmatrix} \right\} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$= \begin{matrix} & a_1 & a_2 \\ c_1 \\ c_2 \end{matrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \qquad (2\text{-}14)$$

すなわち，得られた行列は a_2 と c_2 の関連を示した成分以外は全て 0 の値となっている．式(2-14)の 1 項目には，要素 b と c の関連が示されている．関連を持つのは b_2 と c_2 のみである．その写像の難しさを行列成分として設定し，他の関連は成分値 0 として設定されている．また 2 項目には要素 a と b との関連が示されている．関連を持っていないのは a_1 と b_2 のみであり，行列成分を 0 としている．それ以外は図中に示した写像の難しさを行列の成分値とした設定している．この結果得られた行列の積は，a_2-c_2 の成分のみ 1 を示し，他成分は 0 となっている．つまり，モデル全体の写像の難しさは，a_2 から b_2 を経由して c_2 までの写像経路の難しさのみで決まっており，他の a_1 や b_1 の要素の持つ写像の難しさがモデル全体の難しさに反映されていない．そこで，

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \left\{ \begin{matrix} & b_1 & b_2 \\ c_1 \\ c_2 \end{matrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{matrix} & a_1 & a_2 \\ b_1 \\ b_2 \end{matrix} \begin{pmatrix} 0.5 & 0.5 \\ 0 & 1 \end{pmatrix} \times \begin{matrix} & b_1 & b_2 \\ a_1 \\ a_2 \end{matrix} \begin{pmatrix} 0.5 & 0 \\ 0.5 & 1 \end{pmatrix} \times \begin{matrix} & a_1 & a_2 \\ b_1 \\ b_2 \end{matrix} \begin{pmatrix} 0.5 & 0.5 \\ 0 & 1 \end{pmatrix} \right\} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$= \begin{matrix} & a_1 & a_2 \\ c_1 \\ c_2 \end{matrix} \begin{pmatrix} 0 & 0 \\ 0.25 & 1.25 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

(2-15)

のように，a と b の関係を示す転置行列を式(2-14)に挿入することによって，写像経路を数式化する．その結果，a_1 から c_2 の写像経路の難しさも正しく反映されるようにな

る. 式(2-14)では要素 a から c へ, つまり上から下に向けた写像経路しか表現されない. そのため計算上，b_1 と c_1 の関連がなければ，a_1 から c_1 への写像経路は無いものされてしまう．a_2 から b_1 への経路も，b_1 から c_1 への経路が存在しないために成分値は 0 となる. そこで, a と b の関係を示す転置行列を挿入することで(式(2-15)の左から第 3 項), b から a に向けた下から上に向けた写像経路を行列計算の中に組み込んだのである．これに伴って，転行列によって b から a に向けた写像を再度 a から b に向けた写像にするために式(2-15)の左から第 2 項の行列を挿入している.

同様に写像経路数の算出式は，次式（式(2-16)）となる.

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \left\{ \begin{matrix} c_1 \\ c_2 \end{matrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{matrix} b_1 \\ b_2 \end{matrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \times \begin{matrix} a_1 \\ a_2 \end{matrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \times \begin{matrix} b_1 \\ b_2 \end{matrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \right\} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$= \begin{matrix} \quad a_1 & \quad a_2 \\ c_1 \\ c_2 \end{matrix} \begin{pmatrix} 0 & 0 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

(2-16)

すなわち，式(2-15)における成分値を 0 または 1 に置き換えた行列で構成したものとした．ここで示したように a_1 から c_2 は 1 つの経路のみであり（行列の a_1-c_2 の成分値), 式(15)では成分値が 0.5×0.5×1.0×1.0=0.25 となっている．また，a_2 から c_2 については転置行列を挿入したことで 2 つの経路がある．ひとつは a_2 から b_2 を経由して c_2 を辿る経路である．その場合の計算式は 1.0×1.0=1.0 となる．もうひとつは a_2 から b_1 を経由して a_2 に戻り b_2 を経由して c_2 を辿る経路である．この場合は 0.5×0.5×1.0×1.0=0.25 となる. これらを合計した結果 a_2 から c_2 への経路の成分値は 1.25 のとなる.

２-３-４-４派生パターン 3

派生パターン 3 として，複数プロジェクトを統合する方法を説明する．複数プロジェクトを 1 つの行列として扱うために，行列を小行列化（Submatrix）する．対角成分となる小行列に各プロジェクトのマトリックスを構築する. 非対角成分にある小行列はプロジェクト間の関係を示す．例えば，設計書の再利用や実装部品の再利用などの関係を整理するために用いる（図 2-7）.

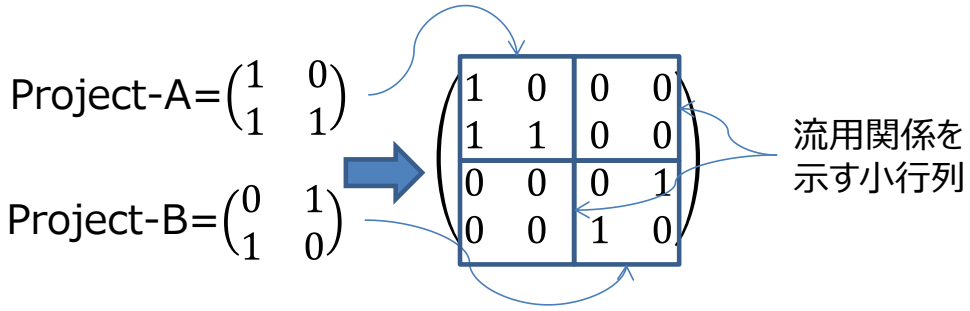


図 2-7　複数プロジェクトの小行列化

Figure 2-7 Submatrix of Projects

例えば，図 2-8 に示したモデルについて説明する．モデル 1 から 4 まで示している．モデル 1 はプロジェクト A とプロジェクト B の間に関連はなく，互いに独立している．つまり，部品の共有などがないことを意味する．モデル 2，3，4 は一部部品を共有している状態を示す．例えば，プロジェクト A で開発した部品 2 をプロジェクト B の部品 1'が利用していることを意図している．モデル 2 は component2 と 1' を，モデル 3，4 は component2 と 2' を共有するケースを示す．モデル 4 はモデル 3 と同様ではあるが，モデル 4 では artifact と component の関係を 1:1 に簡素化している．なお，下記の例では，Feature と Needs は省略している．

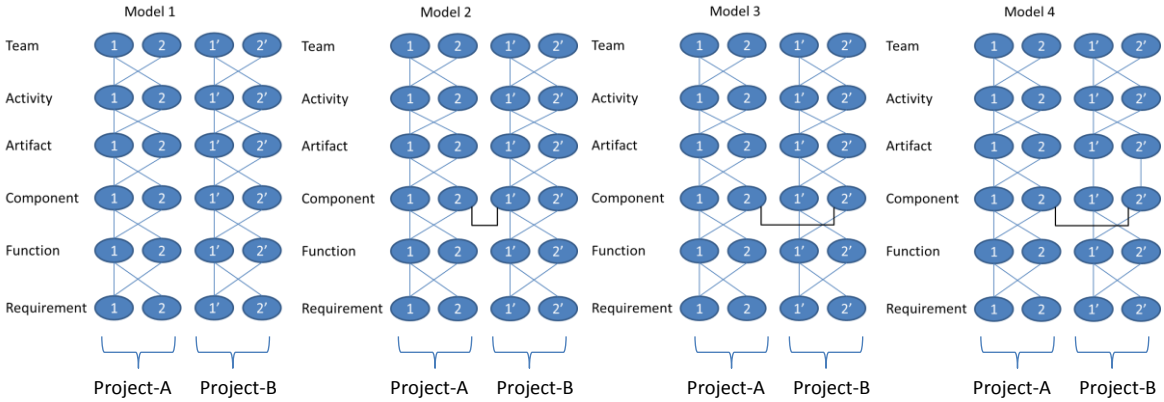


図 2-8 試行ケースで用いたモデル

Figure 2-8 Model of Trial Case

上記の各モデルの複雑性を算出するために，以下（図 2-9）に各プロジェクトの行列化を試みる．行列化するにあたっては，成分値を 1 と設定した例を示す．つまり，写像経路数を算出する例を示す．行列は 4 つの小行列で構成されており，左上の 2×2 の小行列ではプロジェクト A を構成する要素間の関連を設定し，右下の 2×2 の小行列ではプロジェクト B を構成する要素間の関連を設定している．また，左下及び右上の 2×2 の小行列では，プロジェクト A とプロジェクト B の関連を設定している．モデル 2，3，4 において，Componet-Compomemt の行列で色塗りした成分が両プロジェクトの関連付けを示している．なお，モデル 4 において，左下の 2×2 の Artfact-Component の小行列において色塗りしている成分は，プロジェクトを構成する要素間の関連が，他のモデルと異なる箇所を示している．

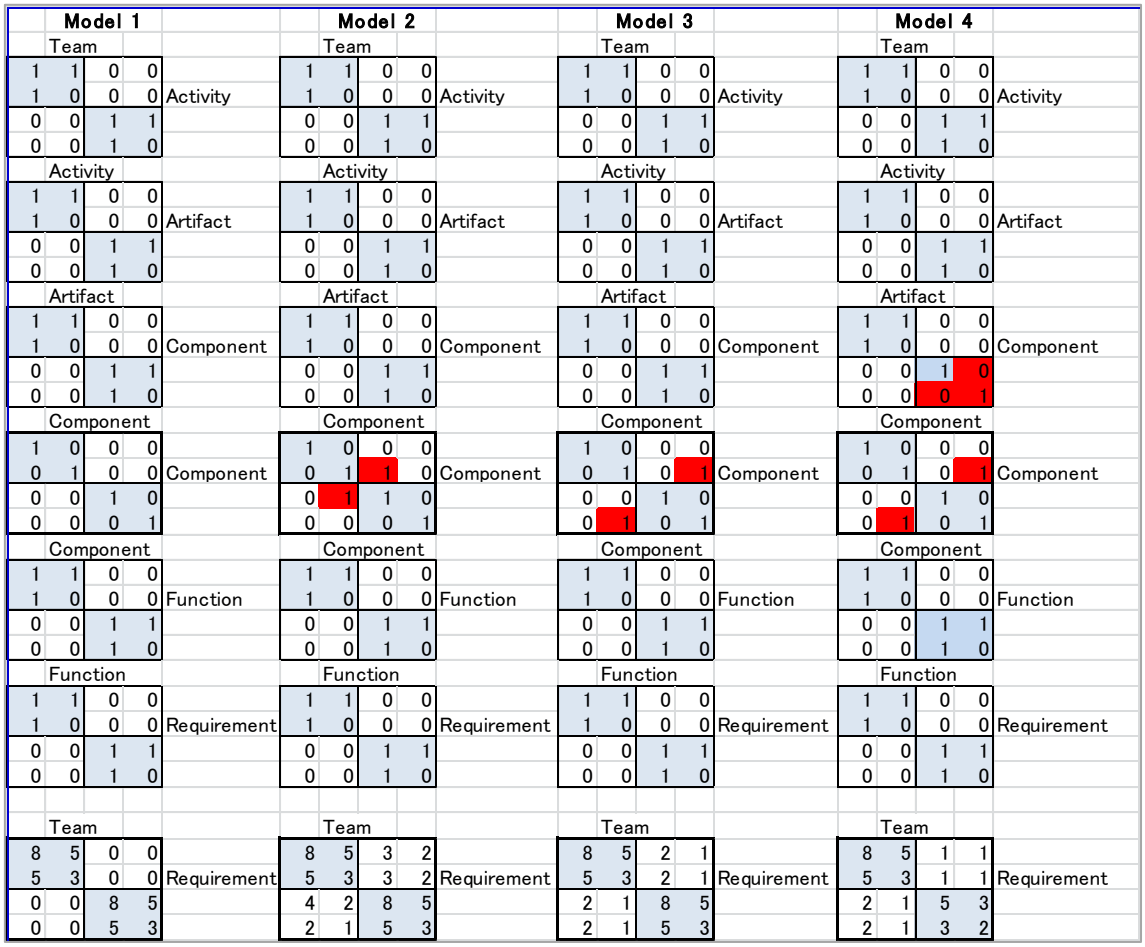


図 2-9 各モデルのマトリクス表現

Figure 2-9 Submatrix of Models

モデル 1 では, 非対角要素の成分値は全て 0 となっている. モデル 2, 3, 4 は Component 間の関係を示す行列の非対角要素に 1 を設定し，部品の共有状態を示している．ここで着目すべき点は，最下部に記載している Team と Requirement の関係を示した行列，つまり，Team から Requirement までに各行列の積，である．すなわち，モデル 1 の対角要素に位置している小行列と, 他のモデルから導かれた対角要素に位置している小行列は，モデル 4 以外，全く同じ成分となっている（モデル 4 はプロジェクト B の形を変形しているため，他のモデルと異なることは当然のことである.）．このように各プロジェクトを対角上にある小行列で表現することで，個々のプロジェクトは，個別にそ

の写像経路を算出することが可能となる．

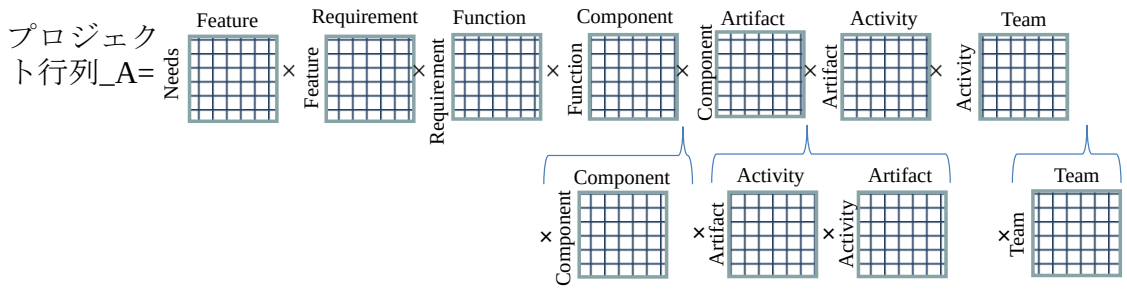
一方，非対角要素に着目してみると，プロジェクト A とプロジェクト B が独立しているモデル 1 では，非対角要素の小行列の成分値は全て 0 である．つまり，当然のことではあるが，互いのプロジェクトを関連付ける写像経路が存在しないことを示している．しかし，モデル 2，3，4 は非対角成分に 0 以外の数値が導出され，プロジェクト A とプロジェクト B を跨った写像経路が出来ていることが示されている．モデル 2 とモデル 3 では共有化している部品が異なるため，両プロジェクトを跨った写像経路数も異なっている．モデル 4 では，プロジェクト A の Team からプロジェクト B の Requirement までの写像経路数（左下の小行列）はモデル 3 と同じである．何故なら，モデル 3，4 のプロジェクト A では，構成する要素間の関連が同じであり，また同時にプロジェクト B の Component，Function，Requirement の関連も同じであるからである．しかし，モデル 4 のプロジェクト B は，Artifact と Component の関連が他のモデルのプロジェクト B と異なり 1:1 となっている．従って，モデル 4 のプロジェクト B の Team からプロジェクト A の Requirement までの写像経路数（右上の小行列）の値は異なったものとなっている．

写像経路の難しさについても，同様に小行列を用いることで，複数のプロジェクトを統合して，算出することが可能である．

2-3-4-5 まとめ

以上のように基本パターンとして式(2-13)と示し，その派生パターンを 3 つ説明した．一般的にここで説明したパターンを用いればプロジェクトの写像経路数や写像の難しさを算出することができると考える．

本研究の実証検証で用いたプロジェクトにおいても，



(2-17)

のように，Team 間の関連や Component 間の関連を表現するために 1 つ目の派生パターンを用い，企画書などの Artifact を介した Activity との写像経路を表現するためには 2 つ目の派生パターンを用いた.
以上をまとめると冗長化したプロジェクトの写像の難しさは(式 2-12)で示され，また，

並行経路数＝行列の次数 (2-18)

冗長な経路数＝|A’－E|=Σ｜a'_{ij}－ e ij｜ (2-19)

となる．但し，行列 A’は行列 A の成分値を 0,又は 1 とした行列であり，E は単位行列である．a_{ij} 又は a'_{ij} は行列 A 及び A’の成分値を示す.

2-4 ソフトウェア開発プロジェクトの複雑性指標の妥当性検証

実際のゲームソフトウェア開発プロジェクトを対象にプロジェクトの複雑性の妥当性検証を行った．妥当性検証にあたっては，ゲームソフトウェア開発プロジェクトをモデル化し，その複雑性と生産性及び開発工数との相関を検証した．

2-4-1 検証プロジェクトの選定とプロジェクトの概要

まず，妥当性検証を行うにはどのようなプロジェクトを選定するべきか，その条件を検討した．そして，①相関検証する上で十分なデータ数を得られること．②プロジェクトを構成する個々の要素の持つ写像の難しさを高い信頼性で数値化できること．そして③複数の開発者がコミュニケーションを取りながら開発プロセスを進め，複数の要素で構成されるプロダクトを生み出していること．以上の 3 つの条件を満たすこととした．そして，これらの条件を満たすプロジェクトとして，日本国内のあるゲーム開発プロジェクトを対象とすることとした．本プロジェクトの開発は 6-7 名で行われており，ゲームソフトウェアという複雑な機能を有する開発を進めていた．また，対象としたのは 21 のプロジェクトであり，相関を検証する上では十分なデータ数と考えた．更に，この 21 のプロジェクトのプロジェクトマネージャーは 1 人であり，(詳細は後述するが) プロジェクトを構成する個々の要素の持つ写像の難しさを高い信頼性で数値化できると考えた．以上のようにはじめに挙げた 3 つの条件を満たしていたことから，本プロジェクトで検証を行うこととした．

プロジェクトの開発期間は 2010 年 12 月から 2011 年 8 月である．この間，プロジェクトでは要求仕様の整理から基本設計，実装，試験という一連の開発工程を実施している．7 プロジェクト毎に開発を行い，顧客への納品と検収を繰り返して 21 プロジェクトの開発を完了した．また，個々のプロジェクトの開発完了の判断は，バグ発生状況が収束傾向にあり，且つバグ抽出件数がある一定値以上に達することを条件とした．なお，今回対象とした 21 プロジェクトの内，No7,14,21 のプロジェクトは図 2-10 の中で Activity としてのアダプタ作成，アダプタ（Artifact）およびアダプタ（Component）の要素が存在するモデルであるが，他の 18 プロジェクトではこれらの要素は存在しない．また同様に，No7,14,21 のプロジェクトでは Team におけるサウンドデザイナーは存在しないが，他の 18 プロジェクトではこの要素は存在するモデルとなる．

2-4-2検証対象プロジェクトのモデル化

対象としたゲーム開発プロジェクトの構成要素を抽出する．プロセス領域・実態領域・機能領域・顧客領域においてどのような要素から構成されているのか把握するために，プロジェクトのディレクター（プロジェクトマネージャーの役割）からヒアリングを行った．その結果を図 2-10 にまとめた．そして，他のソフトウェア開発プロジェクトと非常に類似した要素で構成されていることがわかった. 次節以降で詳細を述べるが, 一般的なプロジェクトで担うべき開発者の役割は網羅されている. またプロセスも前述の通り，基本設計，詳細設計，実装というように一般的なエンジニアリングプロセスの通りである．また，開発されるプロダクトの構造も OS，ミドル，アプリケーションという構成であり, 本プロジェクトが他にも応用可能な検証プロジェクトであると考える. なお，本図では，プロジェクトを構成する要素を UML のクラス図を用いてまとめている．プロジェクトのモデルを作成する上で，必ずしも UML を用いる必要はなく，その他のプレゼンテーションツール等でも作成することは可能である．しかし，UML ツールを用いるとプロジェクトを構成する要素の管理は勿論, 図の作成や描いた要素間の関連を見やすく整理することも容易である. 本図においては, 各クラスに該当するもの（四角の箱内に要素名を記載）が，ヒアリングにより抽出したプロジェクトを構成する要素である．また，要素間を結んだ実線は写像経路を示している．

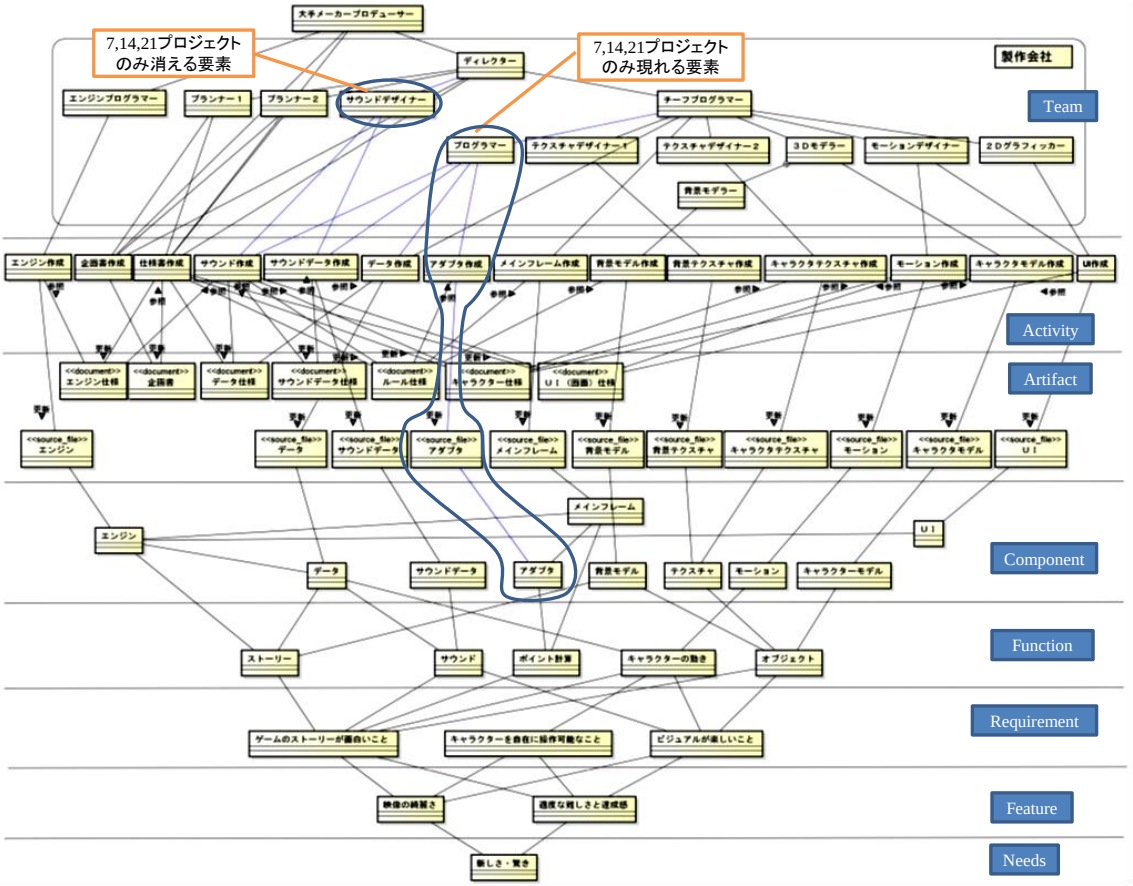


図 2-10　実証検証を行ったプロジェクトの情報写像モデル

Figure 2-10 Information Mapping Model of Validation Project

２-４-２-１プロセス領域の構成要素

まず，プロセス領域の Team の要素を整理するために，組織体制についてディレクターからヒアリングを行った．

プロジェクトマネージャーとしてゲーム開発ではディレクター，ビジネスアナリストとしてプランナー，またアーキテクトとしてチーフプログラマーがおり，そしてプログラマーが存在する．またユーザーインタフェース(以下 UI と記述)デザイナーはサウンドデザイナー，グラフィッカーなどが該当する．このようにプロジェクトの体制を調査し，以下を Team の要素と定義した．

Team；大手メーカープロデューサー，エンジンプログラマー，ディレクター，プランナー，チーフプログラマー，プログラマー，サウンドデザイナー，またグラフィッカーとしてテクスチャデザイナー，モーションデザイナー，2D グラフィッカー，3D モデラー，背景モデラー

次に Team 内の写像経路について述べる．Team 内での写像経路は設計に直接関わらない写像を示す. つまり Activity によって Artifact を作成するための写像以外を示す．例えば，企画書をまとめるために企画書作成という Activity を介した写像経路がある．企画書という Artifact を生成するので, この写像経路は設計作業に直接関連する写像と言える．一方，Activity を介さない Team 内に閉じた写像経路は，例えば進捗管理等の管理上の写像経路を示す. 進捗状況の共有は Artifact を生成する直接的な写像ではない．しかし，間接的には個々の Team の設計作業にも影響を及ぼすため，写像経路としてモデルに組み込むこととした．例えば，他の作業と同期をとるために作業速度を上げないといけない，そのためにある作業の優先度を上げなければならない，といった判断がなされる．これにより，各 Team の有スキル者の配備等の対処がなされ，写像の難しさも変化すると考えたからである．

次に個々の要素間の関連について述べる．図中に示した Team の要素の内，大手メーカープロデューサー以外の要素が実際にソフトウェアを開発する会社の役割を示している．プロジェクトマネージャーの役割はディレクターなので，プランナー，サウンドデザイナー，チーフプログラマーは各自の作業状況はディレクターと共有している．エンジンプログラマーについては, エンジン自体が大手メーカーから提供されていたため, 大手メーカープロデューサーとの関連付けがある．また，チーフプログラマーはプログラマーやゲームに登場するキャラクターやそのルール等を開発するための 3D モデラー, 背景モデラー, モーションデザイナー, テクスチャ, 2D グラフィッカーを率いていた．

なお，アダプタ開発を行うプロジェクトは 7 名の開発要員で開発し，他のプロジェクトは 6 名の開発要員で開発している．要員数の内訳は以下の通りである．ディレクターはプランナーの役割も兼任して 1 名，またもう 1 名のプランナー，サウンドデザイナー，プログラマーの役割は各 1 名で構成した. また背景モデラーと背景テクスチャデザイナー（図中，テクスチャー1）は兼任で 1 名，3D モデラー，キャラクターテクスチャデザイナー，2D グラフィッカー，モーションデザイナーも兼任で 1 名，エンジンプログラマー，チーフプログラマーは兼任で 1 名である．

次に，プロセス領域の Activity の要素を整理するために，開発プロセスをヒアリングした．基本要件を決め，要求仕様書，設計書を作成し，それを基に実装していくというプロセスは，組織体制と同様に他のソフトウェア開発にも非常に類似していた．そのプロセスを調査し，Activity は以下の要素と定義した．

Activity ; エンジン作成，企画書作成，仕様書作成，データ作成，サウンドデータ作成，サウンド作成，UI 作成，またルール作成としてアダプタ作成，メインフレーム作成，背景モデル作成，キャラクターモデル作成，背景テクスチャ作成，キャラクターテクスチャ作成，モーション作成

上記の Activity について留意すべき点は“試験”に関連した要素として挙げていないことである．以下に“試験”を要素として挙げていない理由を述べる．公理的設計論では前述の通り，4 つの領域から成る．このとき，顧客領域は設計課題であり，それに対する設計解は機能領域で示される．同様に機能領域を設計課題と見るならば，設計解は実体領域で示される．このように，図 2-1 においては，下位層に位置する要素が設計課題であり，上位層に位置する要素が設計解なのである．従って，設計という活動は下位層から上位層へ情報が写像される過程である．また試験とは，設計解が正しく設計課題の解に成り得ているかを確認することである．言い換えれば，設計活動とは逆方向（上位層から下位層）へ写像する過程が“試験”である．以上より，設計活動を網羅した要素が抽出できていれば，写像を上下にすることで試験を実施したと見なせることから試験を要素としては挙げていない．すなわち，開発プロセスモデルである V 字モデル[26]の左側の作業のみを挙げている．但し，例えば，試験のために試験担当者を個別に配置し，試験項目表を作成する等の Team の要素がプロジェクト内に存在し，試験項目表といった成果物（artifact）の要素が個別にあるならば，それは情報写像モデルの要素として列挙しなければならない．本研究においては，前述の Team の要素で列挙した担当が，同時に試験担当でもあったことから，試験に関連した要素を挙げていないというこ

とに留意する必要がある.

Activity の写像経路について述べるが, Activity 同士の写像経路は存在しない. Team の各要素が行う Activity を実線で結んでいる．またその Activity から生成される Artifact を実線で結び，その写像経路を整理した.

次に，同様に，成果物を調査し，Artifact を以下のように定義した．図中に示した要素名の上に document と記述したものは日本語や図表等で作成された仕様書・設計書である．同様に source_file と記述した要素はプログラミング言語で作成されたコンパイル前の電子ファイルやデータファイルを示す.

Artifact ; 企画書，仕様書・設計書としてエンジン仕様，データ仕様，ルール仕様，キャラクター仕様，UI（画面）仕様，サウンドデータ仕様，その他エンジン（ソース），データ（ソース），アダプタ（ソース），メインフレーム（ソース），背景モデル（ソース），背景テクスチャ（ソース），キャラクターモデル（ソース），キャラクターテクスチャ（ソース），モーション（ソース），UI（ソース），サウンドデータ（ソース）

また，要素間の関連については，Artifact 同士の関連は無い．従って，上下層との関連のみを実線で示した．企画書作成(Activity)で作成された企画書を別の仕様書作成(Activity）が参照し，エンジン仕様，データ仕様，ルール仕様，キャラクター仕様，UI 仕様を作成していた．サウンドデータ仕様はエンジン仕様やデータ仕様を元にサウンド作成（Activity）を介して作成されている．また，これらの仕様書を元にして上記で挙げた各 source_file が作成される.

２-４-２-２実体領域の構成要素

実体領域を構成する要素を明らかにするために, ゲームソフトウェアのアーキテクチャーを調査し，どのような Component と Function で構成されているのか調査した．ゲームのソフトウェア構造は, 他の企業向けソフトウェアと類似したソフトウェアアーキテクチャーである(図 2-11).

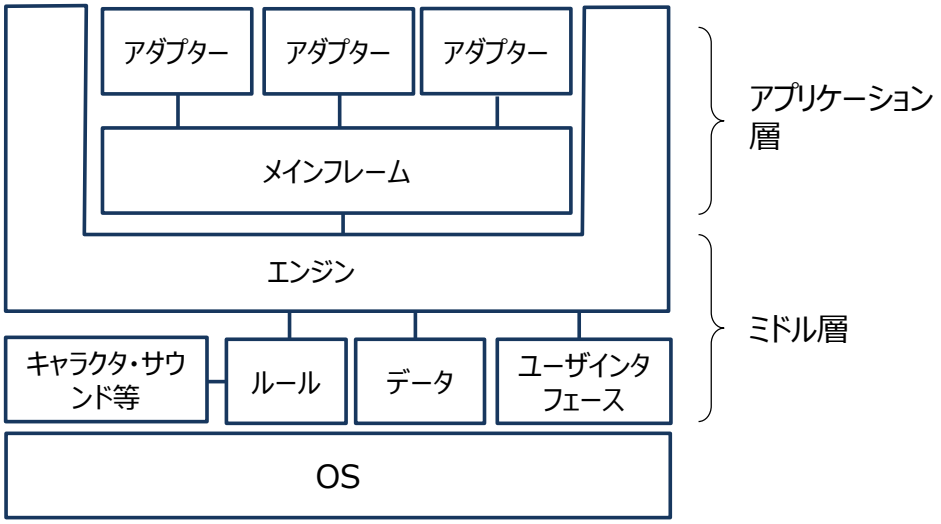


図 2-11　ゲームソフトウェアの構造

Figure 2-11 Game Software Architecture

すなわち，ハードウェア上に OS，その上にエンジンというミドルウェアを積み，ゲームソフトを稼動するという構造になっている．エンジンはゲームに必要な基本機能を保有し，エンジンに対して，ルール，キャラクター，データを定義することでゲームソフトのシナリオが決まる．エンジンの持つ機能だけでは不足している場合，メインフレーム及びそれを補完するアダプタによりエンジンの機能を拡張する．

また，以下のような役割を各要素は担っており，それらが連動してゲームソフトウェアとして機能を提供している．まず，ユーザーインタフェースから入力されたイベントを元にエンジンが動く（エンジンが動く時には，メインフレームやアダプタも必要に応じて起動される）．エンジンはルールによって決められた操作シナリオの制御を行う．そのルールに応じてキャラクターが動作し，その時のキャラクターの動作速度などはデータによって規定されている．ソフトウェアアーキテクチャーとしては，キャラクターやサウンドデータ等は，ルールによりエンジンとは独立性の高い部品となっていた．独立性が高い部品とは，エンジンの開発作業とは無関係にキャラクター等の開発を進められる部品であることを意味している．そのため，図 2-10 ではエンジンと関連付けていない．そしてこれらの部品が統合し，ストーリー，ビジュアル，キャラクターの動き，ボイスデータ，得点のカウントなどの機能を実現している．以上のようにソフトウェア

アーキテクチャーを調査し，Component と Function の要素とその関連を以下に定義した．なお，本プロジェクトでは，図 2-11 で示した要素のひとつであるルールは関連する要素（背景モデル，テクスチャ，モーション，キャラクターモデル，サウンドデータ）の一部として位置付けていた．つまり，例えばキャラクターモデル作成(Activity)の中でルール設定も含めて実行され,キャラクターモデル(Artifact)，キャラクタモデル(component)を作成している．従って，下記の要素には挙げていないが，各要素には包含されている．

Component ;　エンジン，データ，アダプタ，メインフレーム，UI，背景モデル，テクスチャ，モーション，キャラクターモデル，サウンドデータ

Function ;　ストーリー，ポイント計算，キャラクターの動き，オブジェクト，サウンド

２-４-２-３機能領域と顧客領域の構成要素

機能領域と顧客領域の構成要素を整理するために，ゲームソフトウェアに対する顧客の要求がどのようなものなのか調査した．

顧客の要求も他のソフトウェア開発プロジェクトとも類似した構造によって規定されている．ゲーム開発において，ユーザーは，新しさと驚きを感じられるような映像とロールプレイングをゲームで体感すること(新しさ・驚き)を望んでいる(Needs)．その実現のために，基本要件（Feature）として,ゲームソフトウェアは，①ストーリーに即した色彩的な綺麗さと，高い解像度を活かしたデザインを実現したものであること(映像の綺麗さ)と，②ロールプレイ上で，戦略的な駆け引きを必要とした多様なストーリーを実装していること(適度な難しさと達成感)が規定された．従って，要素間については，1 つの Needs から 2 つの Feature に関連付を行った．また，それを実現するために各機能に求められる要求条件（requirement）として，ゲームソフトウェアは，①提供する UI によって，キャラクターの動くスピード，方向を自在に操作可能とするための機能を具備していること(キャラクターを自在に操作可能なこと)，②提供するストーリー展開において，サウンド効果を用いて臨場感を持たせ，キャラクターがシステムと駆け引きすることを可能する．そして，その結果獲得されるポイントによって，ミッションの達成度を判定する機能を具備していること(ゲームのストーリーが面白いこと)，③その映像デザイン・キャラクターの動き・サウンド，これら 3 つの要素を自在に組み合わせ，コントロールすることで，映像デザインからの視覚効果を補強する機能を具備

していること(ビジュアルの楽しさ)が抽出された．これらの関連を整理し図 2-10 に示した．Needs と Feature を顧客領域の要素とし，Requirement は機能要求に等しいので機能領域の要素とした．なお，秘匿義務があるため，各要素の記述は上記のレベルに留めた

以上をまとめ，Needs，Feature，Requirement の各要素とする．

Requirement ; キャラクターを自在に操作可能なこと，ゲームのストーリーが面白いこと，ビジュアルの楽しさ

Feature ; 映像の綺麗さ，適度な難しさと達成感

Needs ;　新しさ・驚き

Needs と Feature を顧客領域の要素とし，Requirement は機能要求に等しいので機能領域の要素とした．

２-４-２-４構成要素の粒度に関する補足

前述のように情報写像モデルの構成要素を決定した．本節では，何故，構成要素をこの粒度にしたのか説明する．

情報写像モデルを構成する要素の粒度は, そのモデル作成及び管理工数とプロジェクトに求められる開発コストの管理精度の間で, そのトレードオフを考慮して決定されるべきである．但し，モデルの可読性を考慮すると，筆者の経験上，各カテゴリの要素数は 10-20 程度に留めることが適切と考える．従って，それ以上の詳細なモデルを作成する場合は，サブシステム分割により，個別にモデル化することが望ましい．その際，プロジェクト全体の複雑性を算出するためには，2-3-4-4 節でも述べたように，その全体を現す行列をブロック（小行列）化し，対角上にあるブロックを各サブシステムに割り当てる．非対角成分ではサブシステム間の写像経路を表現する．このように 1 つの行列をブロック化することで，プロジェクト全体の複雑性を算出することも可能である[19].

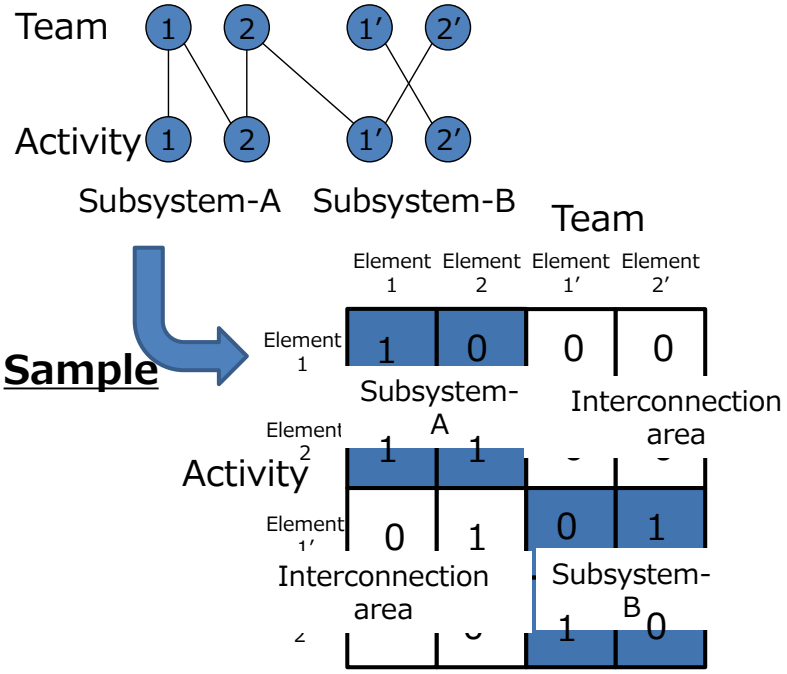


図 2-12 プロジェクトの小行列化表現

Figure 2-12 Submatrix of System

すなわち，図 2-12 に示したように Subsystem-A と B は要素 Team2 と Activity1’で連携しているものとする．その場合，各 Subsystem 内の要素に閉じた関係性は行列の対角上にある小行列に示している．一方，前述の Subsystem 間の連携は非対角上にある小行列（左下のブロック）に示している．Team の要素である Element2 と Activity の Element1’の関連を示す成分として 1 を設定し，このブロック内のそれ以外の成分は 0 を設定し，2 つの Subsystem の関連を示した．また，例えばこの行列の積を算出すれば，対角上にある各 Subsystem のブロックでは，その Subsystem 内の写像経路数が求まる．そして，非対角上にあるブロックでは，Subsystem 間を跨った写像経路数が算出される．このように各要素間の関連性を示す行列として作成し，プロジェクトの写像経路を算出する際に利用することで，要素数の多いプロジェクトでも複雑性を算出することが可能である．

2-4-3検証データの収集

2-4-3-1複雑性指標の算出

複雑性指標を算出するため, 写像経路数と要素の持つ写像の難しさを以下のように規定し，複雑性を算出した.

本検証では，1 人のディレクターが 21 個のプロジェクトそれぞれの開発状況を把握している．まず，抽出した要素で構成する行列の次数により並行経路数を決定し，全写像経路数はプロジェクトの情報写像モデル（図 2-10）より規定した．そして，プロジェクトを構成する各要素の写像の持つ難しさについては 1 人のディレクターに表 2-1 をガイドラインとしてヒアリングし設定した．その結果は表 2-2 の通りである．この結果を元に，2-3-3 で述べた方法により，写像経路の難しさを規定した.これら写像経路数と写像の難しさを規定するにあたっては，2-3 章で述べた計算式に則り，表計算シートによってプロジェクト毎にその複雑性を算出した.

以下，表 2-2 で示した各要素の写像の難しさについて述べる．表中で数値を記入していない箇所はその要素が存在していないことを示している．例えば 7,14,21 以外のプロジェクトではアダプタ作成(Activity)に数値が記入されていないのは前述の通り，プロジェクトで作業が無かったからである．また各要素の傾向については，Team, Activity, Artifact の要素の傾向としては制作が進むにつれて習熟度も向上することから値が小さくなっていく傾向がみられる．しかし，Component や Function についてはプロジェクトによって数値の大小があるが, それはプロジェクトの仕様の違いから生じているものであった．Requirement, Feature, Needs についてはアダプタ開発の有無によって違いはあるものの，その他のプロジェクトは全く同じ値を示している.

表 2-2 実証検証を行ったプロジェクトの要素とその写像の難しさ

Table 2-2 Difficulty of Mapping for Project Elements

		プロジェクト																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
team	大手メーカープロデューサー	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	大手メーカー技術部門	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	ディレクター	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.1	0.1	0.1
	プランナー1	1.9	1.9	1.9	1.8	1.7	1.6	1.9	1.5	1.5	1.4	1.3	1.2	1.1	1.3	1.2	1.1	1	1	1	1	1
	プランナー2	1.9	1.9	1.9	1.8	1.7	1.6	1.9	1.5	1.5	1.4	1.3	1.2	1.1	1.4	1.3	1.3	1.3	1.2	1.2	1.2	1.1
	チーフプログラマー	0.7	0.7	0.6	0.6	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.4	0.4	0.5	0.3	0.3	0.3	0.3	0.2	0.2	0.1
	プログラマー	–	–	–	–	–	–	1.2	–	–	–	–	–	–	1	–	–	–	–	–	–	1.5
	テクスチャデザイナー1	1.1	1.1	1.1	1.1	1	1	1	0.8	0.8	0.7	0.7	0.6	0.6	0.5	0.3	0.3	0.3	0.2	0.2	0.1	0.1
	テクスチャデザイナー2	1.1	1.1	1.1	1	1	1	1	0.9	0.8	0.7	0.7	0.6	0.5	0.5	0.3	0.3	0.3	0.3	0.3	0.3	0.2
	モーションデザイナー	0.6	0.6	0.6	0.5	0.5	0.5	0.5	0.4	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	2Dグラフィッカー	1	1	1	1	1	1	1	0.8	0.8	0.7	0.6	0.5	0.5	0.5	0.5	0.5	0.4	0.4	0.4	0.4	0.4
	3Dモデラー	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.7	0.7	0.6	0.5	0.5	0.4	0.4
	背景モデラー	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.8	0.7	0.7	0.6	0.5	0.5	0.5
	サウンドデザイナー	1.7	1.7	1.6	1.6	1.5	1.5	–	1.4	1.4	1.3	1	1	1	–	1	1	1	1	1	1	–
activity	エンジン作成	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	企画書作成	1.1	1.1	1	0.9	0.9	1	0.8	0.8	0.8	0.8	0.7	0.8	0.8	1	0.8	0.8	0.9	0.7	0.7	0.8	0.6
	仕様書作成	1.7	1.7	1.3	1.5	1.3	1.5	1.3	1.1	1.1	1.4	1.4	1.7	1.4	1.2	1.1	1.1	0.7	1.4	0.7	1.1	1.1
	データ作成	1.9	1.6	1.2	1.9	1.6	1.9	0.7	1.6	1.2	1.2	1.6	1.9	1.6	0.9	0.8	0.8	0.8	1.2	0.8	0.8	0.5
	アダプタ作成	–	–	–	–	–	–	1.9	–	–	–	–	–	–	1.2	–	–	–	–	–	–	1.9
	メインフレーム作成	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	背景モデル作成	1.9	1.9	1.2	1.6	1.2	1.2	0.8	1.2	1.2	1.5	1.6	1.9	1.6	1.6	1.2	1.3	1.2	1.6	1.2	1.6	0.1
	キャラクタモデル作成	1.9	1.9	1.6	1.9	1.6	1.6	0.9	1.5	1.7	1.6	1.7	1.7	1.9	1.2	1.2	1.2	1.5	1.3	1.3	1.5	0.4
	背景テクスチャ作成	1.3	1.5	1	1.3	1.1	0.9	0.6	0.9	1	1.1	1.1	1.2	1.3	0.9	0.8	0.9	0.9	1.2	0.9	1	0.1
	キャラクタテクスチャ作成	1.9	1.9	1.2	1.7	1.6	1.8	0.8	1.5	1.6	1.5	1.7	1.5	1.9	1.4	1.2	1.2	1.3	1.6	1.2	1.6	0.4
	モーション作成	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5	0.5	0.5	0.5	0.5	0.5	0.8
	UI作成	0.3	0.3	0.3	0.3	0.3	0.3	0.5	0.3	0.3	0.3	0.3	0.3	0.3	0.6	0.3	0.3	0.3	0.3	0.3	0.3	0.4
	サウンドデータ作成	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.2
	サウンド作成	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.4	0.4	0.4	0.4	0.4	0.4	0.1	0.3	0.3	0.3	0.3	0.3	0.3	0.1
artifact	企画書	0.5	0.5	0.4	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	エンジン仕様	1.2	1.2	1.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	データ仕様	1.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	ルール仕様	1.7	1.7	1.6	1.6	1.5	1.5	1.5	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
	キャラクター仕様	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	UI（画面）仕様	0.4	0.4	0.3	0.3	0.3	0.3	0.6	0.2	0.2	0.2	0.2	0.2	0.2	0.5	0.2	0.2	0.2	0.2	0.2	0.2	0.5
	サウンドデータ仕様	0.4	0.4	0.3	0.3	0.3	0.3	0.5	0.2	0.2	0.2	0.2	0.2	0.2	0.5	0.2	0.2	0.2	0.2	0.2	0.2	0.5
	エンジン（ソース）	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	データ（ソース）	0.7	0.7	0.6	0.6	0.6	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	アダプタ（ソース）	–	–	–	–	–	–	1.9	–	–	–	–	–	–	1.9	–	–	–	–	–	–	1.9
	メインフレーム（ソース）	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	背景モデル（ソース）	1.6	1.6	1.5	1.5	1.5	1.5	1.5	1.5	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
	背景テクスチャ（ソース）	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
	キャラクタモデル（ソース）	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	キャラクタテクスチャ（ソース）	1.6	1.6	1.5	1.5	1.5	1.5	1.4	14	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
	モーション（ソース）	0.5	0.5	0.5	0.5	0.5	0.5	0.6	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.5
	UI（ソース）	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	サウンドデータ（ソース）	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2
component	エンジン	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7
	データ	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
	アダプタ	–	–	–	–	–	–	1.5	–	–	–	–	–	–	1.5	–	–	–	–	–	–	1.5
	メインフレーム	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	背景モデル	1.9	1.9	1.9	1.9	1.9	1.9	1	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1
	テクスチャ	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
	モーション	0.5	0.5	0.5	0.5	0.5	0.5	1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1
	キャラクターモデル	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
	UI	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	サウンドデータ	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
function	ステージ	1.5	1.5	1.5	1.5	1.5	1.5	0.5	1.5	1.5	1.5	1.5	1.5	1.5	0.5	1.5	1.5	1.5	1.5	1.5	1.5	0.5
	ストーリー	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	キャラクターの動き	1	1	1	1	1	1	1.3	1	1	1	1	1	1	1.3	1	1	1	1	1	1	1.2
	オブジェクト	1.5	1.5	1.5	1.5	1.5	1.5	1	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	0.6
	サウンド	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
requiremer	キャラクタを自在に操作可能	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	ゲームのステージが面白い	1.9	1.9	1.9	1.9	1.9	1.9	1.5	1.9	1.9	1.9	1.9	1.9	1.9	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
	ビジュアルの楽しさ	1	1	1	1	1	1	0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	0.5
feature	映像の綺麗さ	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
	適度な難しさと達成感	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
needs	新しさ・驚き	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9

２-４-３-２開発工数データの収集と平均作業日数の算出

プロジェクト毎に各作業の稼動工数データを収集した. 収集した稼動工数をプロジェクト単位に合計し，それを開発工数とした．また開発工数を各プロジェクトの要員数で割ることによって 1 人あたりの平均作業日数を算出した．なお，大手メーカープロデューサーは発注側であるため，開発要員数及び開発工数のデータには含めていない.

２-４-３-３収集データの信頼性向上のための方策

写像の難しさの数値化は前述の通り 1 人のディレクターにより実施した. 何故ならば, 複数人で設定値を決める場合，その設定基準の認識が完全に合わない限り，設定値の大小関係を可能な限り正確に得ることが困難だからである.そこで本研究では, 1 人の設定者が複数の要素を相対評価することによって, 設定者が異なることによる主観の違いから生まれるノイズを排除した. 従って, 21 個のプロジェクトを熟知し評価可能である 1 人のディレクターによって，その数値設定を行うこととした.

２-４-４実証検証

２-４-４-１検証の進め方

まず，複雑性と 1 人あたりの平均作業日数の相関を確認する．式(2-2)の検証により式(2-1)及び(2-3)の妥当性を確認する.

次，複雑性と工数の相関を確認する．式(2-8')を検証することで式(2-8)の妥当性を確認する．式(2-3)から式(2-5)を算出し，その結果を元に式(2-4)を求める．その結果から式(2-8)の計算を行い，式(2-8')の検証を行う.

２-４-４-２複雑性と生産性（1 人あたりの平均作業日数）の相関

複雑性を x 軸，1 人あたりの平均作業日数を y 軸として，検証の結果を以下に示す.

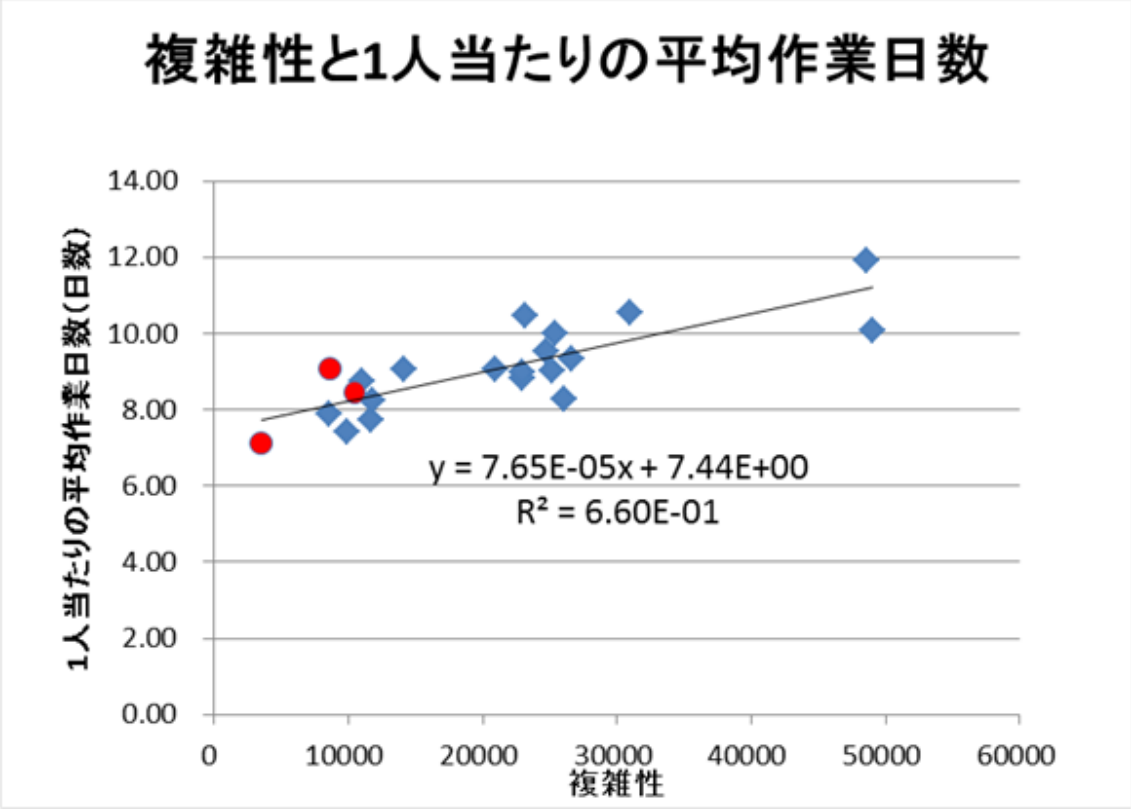


図 2-13　複雑性と 1 人あたりの平均作業日数の相関

Figure 2-13 Graph Relating Complexity to Average Work Days per Person

すなわち，図 2-13 により，プロジェクトの複雑性は 1 人あたりの平均作業日数と非常に良く相関していることがわかる．相関関係は

$$y=7.65E\text{-}05x+7.44 \tag{2-20}$$

として表される．相関係数は 0.813 (有意確率< 0.001)である．

以上のように, 提案した複雑性指標と 1 人あたりの平均作業日数には強い相関がみられた．図中の 3 つの丸印のデータはアダプタ開発を行ったプロジェクトであり，他の 18 個の四角印のデータはそれ以外のサウンドデザイナーがいないプロジェクトのデータである．プロジェクトの体制が異なるにも関わらず，相関係数が示している通りに 21 のプロジェクトが非常に良く近似直線上に乗っていることが分かる．以上により，

プロジェクトの複雑性指標がプロジェクトの生産性をマネジメントする上で有用であることを検証した.

２-４-４-３複雑性指標の総和と開発要員数の積と開発工数の相関

複雑性指標の総和と開発要員数の積を x 軸，開発工数を y 軸として，検証結果を以下に示す.

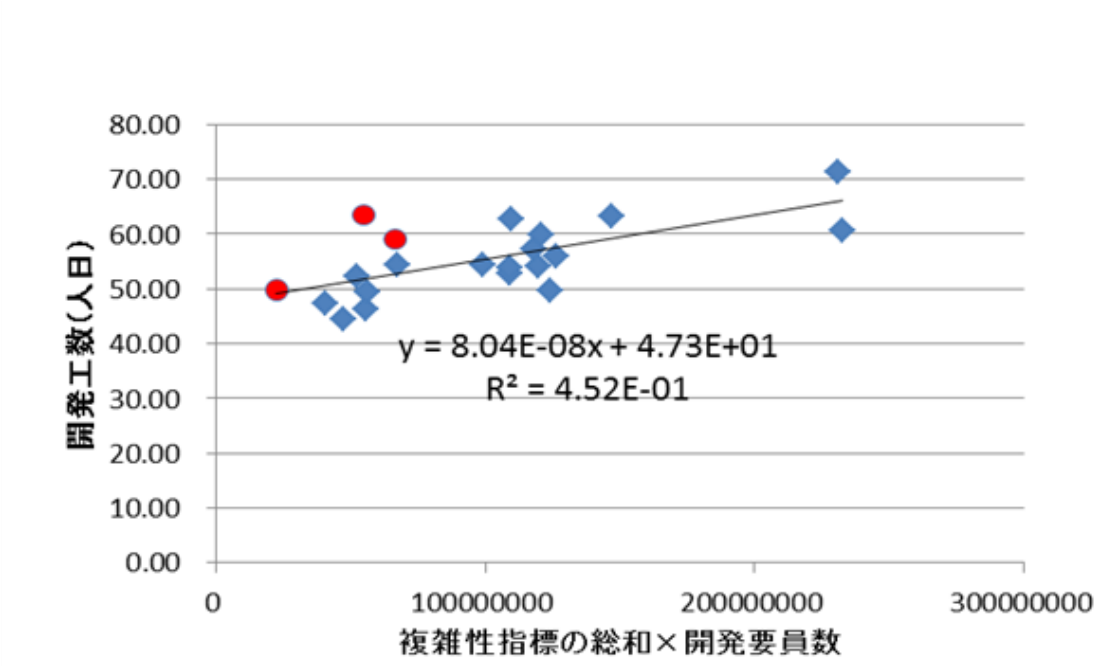


図 2-14　複雑性指標の総和と開発工数の相関

Figure 2-14 Graph Relating Work of Complexity to Man-hour

すなわち，図 2-14 より，複雑性指標の総和と開発要員数の積と，開発工数は強く相関（相関係数は 0.672 (有意確率< 0.001)）していることがわかる．なお，近似線は

$$y=8.04E\text{-}08x+4.73E+01 \qquad (2\text{-}21)$$

である．従って，式(2-8')が確認されたので，式(2-8)は成立すると考える．その結果，プロジェクトの複雑性のマネジメントは，開発工数の低減に効果があることがわかり，マネジメント指標として有用であることを確認した．

次に，式(2-8)による効果について述べる．2-2-7 節で述べたように，式(2-2)から，複雑性と要員数の積も，開発工数と相関すると考えられている．下図に，複雑性と開発要員数の積を x 軸，開発工数を y 軸として，その相関を検証したグラフを示す．

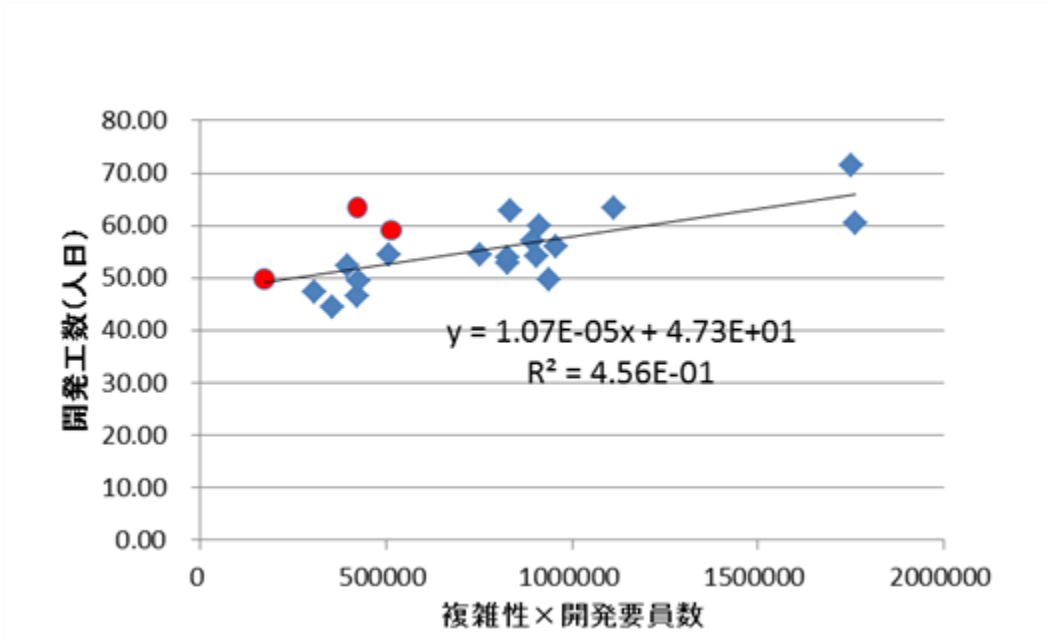


図 2-15 複雑性と開発工数

Figure 2-15 Graph Relating Complexity to Man-hour

すなわち，図 2-15 より，複雑性と要員数の積と，開発工数も強く相関（相関係数は 0.675 (有意確率< 0.001)）していることがわかる．なお，近似線は

$$y=1.07E\text{-}05x +4.73E+01 \quad (2\text{-}22)$$

である．

図 2-14 と図 2-15 のグラフを比較すると，相関係数は 0.672 と 0.675 と大差無く，有意確率も互いに 0.001 以下であった．つまり，式(2-8)を用い，経路数に応じた複雑性を積み上げて開発工数を算出しても，プロジェクトに存在する全経路の複雑性から開発工数を算出しても変わりがないこととなる．相関係数が 0.67 を示していることから概算で開発工数を算出する上では，どちらの方法でも算出結果に差異はないと考える．しかし，今回の検証では，式(2-8)における α_j の精度に課題があると考える．その改善により，今後更なる相関係数の向上が見込める．その課題とは，式(2-8)の導出時に，プロジェクトの全開発期間を通して要員数を一定とし，冗長な経路数を 0 本の時から最大値 n 本まで各 α_j との積を求め，その総和を算出している点である．何故なら，実際の写像経路数の変化は，例えば，全要員が関わって 0 本から n 本まで変化したわけでは無く（つまり，その間の α_j はゼロ），ある本数(x)から n 本までの限定された変化だった可能性が考えられるからである．その場合，0 から x 本までの間に積み上げた複雑性の和は，本来生じていないものであり，誤差の要因と成り得る．このように α_j のデータ精度を改善することにより，式(2-8)の精度向上が見込めると考えられる．

2-4-4-4 相関に関する補足データ

本研究で提案した複雑性指標を算出する上では，プロジェクトの様々な要素を元にしている．個々の要素自体も当然，生産性や開発工数に効く要因を持っている．従って，本節では，複雑性と生産性や開発工数との相関が，ある特定の要素に起因したものなのか否かを確認する．表 2-2 で示したプロジェクトの個々の要素の持つ写像の難しさと，生産性や工数との相関についてまとめたデータを表 2-3 に示す．

表 2-3　各要素カテゴリと生産性，又は工数との相関する要素数の状況

Table 2-3 Assessment of the Correlation

要素のカテゴリ	平均作業日数との相関			工数との相関		
	○	△	×	○	△	×
Team	5	6	3	5	6	3
Activity	7	1	6	1	6	7
Artifact	6	2	10	4	7	7
Component	0	0	10	0	0	10
Function	0	0	5	0	0	5
Requirement	1	0	2	0	1	2
Feature	0	0	2	0	0	2
Needs	0	0	1	0	0	1
小計	19	9	39	10	20	37
合計		**67**			**67**	

相関係数の値による各カテゴリの定義

・強い相関（○）；0.6 以上

・弱い相関（△）；0.4 以上 0.6 未満

・相関なし（×）；0.4 未満

すなわち，ある特定の要素が相関要因となっている訳ではない．例えば，Team のカテゴリに存在する要素の内，1 人あたりの平均作業日数と相関係数 0.6 以上の強い相関があるものは 5 つしかない．弱い相関は 6 つ，相関が無い要素は 3 つである．このように，図 2-10 で示した各要素のカテゴリが同一であっても，相関するか否かは要素次第である．また相関する要素に限定してその特性を調査しても，共通する特定の性質（例えば多くの経路が通っている等）は発見されなかった．従って，生産性や工数と相関した指標として，個々の要素の相関を個別に確認し活用するよりも，本研究で示した複雑性指標を用いる方が有効と考える．

2-5 ソフトウェア開発プロジェクトにおける写像経路の影響についての考察

検証の結果, 提示したモデルの複雑性と生産性や開発工数と相関することがわかった. その複雑性は写像経路を変数としているが, 生産性や開発工数と写像経路の関連について，まだ述べられていない．そこで，本章では，その複雑性を生じさせている写像経路と生産性や開発工数にはどのような関連があるのか考察を行う.

2-5-1 設計プロセス及びコミュニケーションに及ぼす写像経路の影響分析

一般的には生産性向上や開発工数の低減は, 設計情報の写像という観点よりも設計プロセスの観点で考えられることが多い．例えば，生産性向上のためにソースコードの自動生成ツールを用いることで実装プロセスを省略できる．それによって，プロジェクトの生産性が向上し開発工数が低減されると考えられている. そこで本節では設計プロセスに対して写像経路がどのように影響を及ぼすのか, 特にコミュニケーションの観点を含めて，その関連を明らかにする.

設計プロセスにおいて，個々の開発者が設計解を得るために 2 つの手順を踏む．①設計条件を整理すること．②その条件を満たすための解を複数案検討し，その最善案を選択すること．これらの 2 つの手順が，設計解を得るために必要である．ソフトウェアを構成する部品間で冗長な経路が有る場合, ①の設計条件を整理する際に互いの条件を擦り合わせること，すなわち，コミュニケーションが必要となる．例えば，インタフェースすべきデータ項目の調整などがそれにあたる．図 2-1 に当てはめると，部品（Component）間の連携がある場合には開発者（Team）間で作業（コミュニケーション）を通して部品（Component）間のインタフェースを規定する．その規定した内容は設計書（Artifact）に反映され，最後に部品（Component）にも実装されることと解釈される．このプロセスは，複雑性指標の概念においては要素間の相互依存性を下げること, つまり冗長な経路数を低減することによって生産性向上や開発工数の低減が可能である．そして，インタフェースすべきデータ項目が決まってしまえば，それ以降の手順は個々の開発者に閉じた作業となる. つまり, ②の作業は個々の開発者の作業となり, その作業に開発者間に複数の冗長な写像経路が生じていない．従って，他者との調整作業（コミュニケーション）をせずとも，個々の開発者の写像能力が高ければ開発コストは下がるのである. このように設計プロセスの観点からも複雑性指標と生産性向上と開発工数低減との相関を理解することが出来る. 特に, 冗長な写像経路を低減することは, 設計プロセスとして行われるコミュニケーションの軽減につながることが理解される.

2-5-2写像経路のマネジメントによる工数低減及び生産性向上策の検討

本節では，図 2-4 で示した生産性，複雑性と冗長な経路数の関係を用いて，生産性の向上や開発工数の低減のために，写像経路をどのようなマネジメントすべきなのかについて述べる．

前節で示された図 2-4 により図の物理的な意味を考察すると，工数低減・生産性向上には①1 本平均の写像の難しさを小さくすること，②プロジェクトを構成する要素数を少なくすること，③冗長な写像経路を少なくすることが必要であることが有効であることが分かった．何故ならば，生産性向上には，グラフの傾きの低減，y 切片の低減が必要だからである．グラフの傾きの低減には，1 本平均の写像の難しさを小さくすることが必要である．また y 切片の低減には基本プロジェクトの写像の難しさを小さくすることが必要である．それにはグラフの傾きの低減と同様に 1 本平均の写像の難しさを小さくすること，または，並行経路数を小さくすることが必要である．並行経路数を小さくすることとは行列の次数を小さくすることに等しい．それは，生産性向上にはプロジェクトを構成する要素を極力少なくすることと同意である．また，開発工数の低減には，方程式で構成される図形の面積を小さくすることが必要である．従って，生産性と同様に傾きを小さくするために 1 本平均の写像の難しさを小さくすること，そして y 切片を小さくすることが必要である．また冗長経路を低減することも台形の面積を小さくする上で効果が望める．なお，当然のことではあるが，開発工数は上記の面積と開発要員数の積である．従って，そもそも開発要員数を少なくすることも開発工数を低減するためには有用である．

2-5-3写像経路の冗長性の低減によるプロジェクト生産性向上と開発工数低減の可能性

前節で述べた方策が実際にどの程度プロジェクトにインパクトを与えるのか，冗長な経路数を x 軸に，1 人あたりの平均作業日数を y 軸にとって，以下にその試算結果を示す．

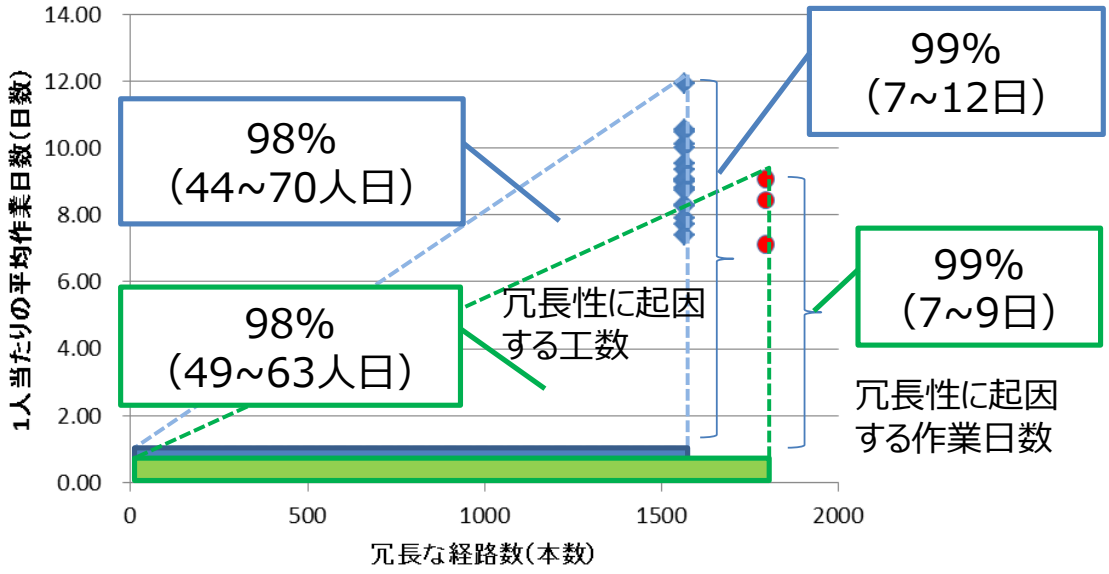


図 2-16　検証対象プロジェクトにおける冗長な写像経路起因の生産性と開発工数

Figure 2-16 Productivity, Man-hour and Redundant Path on the Game Software Development Project

すなわち，本検証によって更なる工数低減や生産性向上を可能とする潜在的な領域の存在が確認された．前述のグラフと同様に，図中の 3 つの丸印のデータはアダプタ開発を行ったプロジェクトであり，他の 18 個の四角印のデータはそれ以外のサウンドデザイナーがいないプロジェクトのデータである．どちらのプロジェクトも，図 2-16 に示すように点線で囲った上三角形が冗長な経路数に起因した平均作業日数や開発工数を示している．上三角形の下に位置する長方形で示された部分は，冗長性がなかった場合の平均作業日数や開発工数を示す．このように冗長性に起因した割合は非常に高い．なお，試算した結果によると，本検証プロジェクトでは写像経路 1 本あたりの平均作業工数＝0.03～0.05 人日／本になることが分かった．

但し，現状このような冗長経路数に起因した工数の内，本当に冗長な工数を判断する基準は確立できていない．例えば，部品(Component)間の構造が，技術的な問題から図 2-1 の左図に示したような相互依存関係を構築することが困難な場合もある．その技術的な問題を解消することで図 2-1 の左図のモデルに近づける方法と，部品構造を修正す

ることはせずに他の要素間の相互依存を低減する方法と，例えばこのような 2 つのアプローチが考えられる．この 2 つの選択肢の内，どちらを選択するかによっても何を冗長な経路とするのか，その基準が異なる．従って，その選択は前節で述べた方策を元に現実解を個々のプロジェクトで検討し，プロジェクトにおけるその冗長性の削減に取り組む必要がある．

2-5-4 複雑性指標の有用性

2-4 章，及び 2-5 章の前節までで示した実証検証の結果とその考察から，本研究で提案した複雑性指標を用いることの有用性について述べる．

2-4-4-2 節と 2-4-4-3 節で示した検証結果から，複雑性指標と生産性及び開発工数が相関することを示した．更に，2-5-1 節の考察で，複雑性と生産性及び開発工数の因果関係を説明した．すなわち，複雑性の低減を図ることで，プロジェクトの生産性向上や，開発工数の低減が可能となることを示した．例えば，プロジェクト開始時に，情報写像モデルを作成し，プロジェクトのメンバーで，そのモデル構成について検討を行う．そして，モデルの複雑性を低減するようなプロジェクト計画を立案する．その結果，プロジェクト完了後の実績として，生産性の向上や開発工数の低減が期待できる．また更に，派生効果として，品質の向上も見込まれる．何故ならば，2-5-1 節で述べたように，個々の開発者の考える時間，すなわち，設計条件を満たす解を複数案検討し，その最善案を選択するための時間を確保できるようになるからである．

現状，生産性の向上や開発工数の低減のために，例えば，ソースコードの自動生成ツールなどの方策が多くのプロジェクトで活用されている．しかし，導入したツールによって，写像経路の難しさはどれだけ低減するのか，関連する写像経路の写像の難しさはプロジェクト全体にどれだけ効くのか，何本の写像経路に影響するのか等は十分に検討されている訳ではない．2-4-4-4 節で述べたように，個々の要素の難しさをコントロールするだけでは，必ずしも生産性が向上するとは限らず，開発工数が低減するとも限らない．すなわち，単に開発者が作成するソースコード量を低減することのみに着眼した自動生成ツールの導入では，必ずしもその効果によって生産性向上や開発工数の低減が実現される訳ではない．そのようなツール導入では，逆に導入したツールの購入費やライセンス費だけ開発費が増大してしまうこととなる．一方，本研究で示した情報写像モデルによって写像経路を簡素化し，複雑性を低減することによって，生産性や開発工数を改善することは 2-4 章で示した通りである．更に，2-5-3 節で述べたように，複雑性

指標に着目することによって，プロジェクトには，生産性や開発工数を大きく改善する潜在的な領域が残されていることが示されている．

以上から，現在のソフトウェア開発の状況に鑑みても，本研究で示した情報写像モデルによって写像経路を簡素化し，複雑性を低減することは，ソフトウェア開発プロジェクトの生産性を向上させ，開発工数を低減するために有用であると考える．

2-6まとめ

2-6-1まとめ

本研究では，ソフトウェア開発プロジェクトの要素間の複雑な関係をシンプルに表すモデルとその複雑性指標を提示し，その有用性を検証した．まず，公理的設計論を元にしたプロジェクトモデルにより，プロジェクト内の設計情報の写像過程の指標として複雑性を定義した．その複雑性は，その写像経路の難しさと写像経路本数に依存する方程式で定義した．写像の難しさや写像経路本数は，プロジェクトモデルを構成する要素関係を表す行列により求める方法を提示した．そして，ゲームソフト開発プロジェクトを元に上記の提案を検証した．検証の結果，提案した複雑性はプロジェクトの生産性（1人あたりの平均作業日数）に強い相関があることが分かった．従って，行列を用いた写像経路の難しさを算出する方法の妥当性を示す事ができた．また同様に，各冗長な経路の複雑性値の総和とプロジェクトの開発工数との相関を検証し，強い相関があることを確認した．以上から，定義した複雑性や複雑性指標の総和を示す方程式の妥当性を確認した．これらの検証結果から，生産性（1人あたりの平均作業日数）や開発工数の大半が冗長性経路により生じていることを試算し，また同時にその改善方法を示し，複雑性指標の有用性について述べた．

2-6-2今後に向けて

本手法を持って，以下の3点の取り組むことにより，プロジェクトの冗長性を削減し，生産性向上・工数削減に対して大きな効果が期待できると考えられる．

①開発者同士のコミュニケーションが極力少なく済むようにすること（冗長性を低減するため）．

②開発者にとって極力容易な要求条件の開発に従事させること（写像を容易にするため）．

③開発者数や作業数，部品数は極力少なくなるようなプロジェクト設計を行うこと（マトリックスの次数を低減するため）．

更にそのためには，①各部品・機能間で極力インタフェースが少なくなるような要求仕様の整理をすることが必要である．②開発すべき部品・機能は開発者のスキルを考慮した分担にしなければならない．言い換えれば，要求仕様と開発者のスキルを考慮した部品構成であることが必要である．③そして，構成する部品数，開発者数が最小化する方法を選択することが必要である．これらを満たすためにプロジェクト計画時から本研究で提案した指標を活用し，より簡素化を図ったプロジェクトの体制・プロセス等を検討することが有用である．本指標の活用によって，単に設計作業が簡素化するだけでな

く，その周辺作業，例えば成果物の構成管理（版数管理）も容易になる．複雑性が低くなれば各成果物は個々の作業と関連性が薄くなり，ある作業の手戻りによって改版された設計書が他の作業に影響を及ぼすことも少なくなるからである．従って，構成管理誤りに伴うトラブルの軽減や，各作業間の同期合わせ等の作業スケジュールの調整稼動も軽減されると考えられる．このように本複雑性指標を活用することによって，ソフトウェア開発プロジェクトの効率化の実現に貢献できれば幸いである．

次章では，本研究結果により提案した複雑性指標が工数と工期の関係を決定付ける要素であることを述べる．具体的には工数と工期の関係式を構成する係数が，本複雑性指標により決定されていることを示す．この結果，上記でも述べたように本複雑性指標によって，効率的な開発，すなわち，同じ工数でもより短い工期で開発すること，又は同じ工期でもより工数を少なく開発することが可能であることを理論的に示す．

【第３章】
複雑性指標に基づく
工数と工期の理論式

３．　複雑性指標に基づく工数－工期の理論式

3-1 プロジェクトの工数と工期の関係

ソフトウェア開発プロジェクトの工数（プロジェクトの各作業に関わる要員数とその作業期間の積の総和（人月））と工期（プロジェクトの開発開始から終了までの期間）の関係は，実証データを元にした経験モデルとして，累乗関数（工期=a×工数^{b}）で表せることが知られている．しかし，その累乗関数において，乗数 a，指数 b は先行研究[1][2][3][4][5][6][7][8]によって様々である（表 3-1）．この様々となる理由は，環境差であると言われているが[9][10]，具体的にどのように何が影響し，乗数・指数を決めるのか，その因果関係は明らかではない．COCOMO[1][2]等では，乗数，指数の決定要因を挙げているが，諸説色々とあり[9]，明らかになったとは言い難い．一方，理論モデルとしては Putnam モデル[11]が有名であるが，プロジェクトの工数や工期に関連した研究は，他に Design Structure Matrix (DSM)[12] を応用した研究[13][14[15][16][17][18][19][20] や，System Dynamics[21] を応用した研究[22][23][24][25][26][27][28][29][30][31]などがある．しかし，それらの理論モデルは経験則にマッチしないなどの問題がある．

表 3-1　各経験モデルの乗数・指数

Table 3-1 Multiplier and Exponent of Models

研究事例			乗数	指数
COCOMO Ⅰ[1]	Organic		2.5	0.38
	Semi-detached		2.5	0.35
	Embedded		2.5	0.32
COCOMOⅡ[2]			3.67~5.25	0.28~0.34
C. E. Walston, C. P. Felix [3]			2.47	0.35
D. Reifer[4]	Web-based electronic commerce		2.0	0.5 (0.32 for larger)
	Financial/trading application		2.2	
	Business-to-business application		1.5	
	Web- based portals		1.8	
	Web-based information utilities		2.0	
ソフトウェア開発データ白書 2012-2013[5]		新規開発	–	0.32
		改良開発	–	0.31
ソフトウェアメトリクス報告 2011年度版[6]			2.54	0.33
経済調査研究所研究レポート 2010[7]		新規開発	2.44	0.34
		改良開発	1.73	0.39
戸田，松本，押野，高橋[8]	成功プロジェクト事例		2.38	0.33
	失敗プロジェクト事例		2.94	0.27
	Javaによる成功事例		1.64	0.42

また，経験モデルを理論的に説明することが出来ていない．つまり，理論モデルと経験モデルの間で，整合性がとれていない．この問題を解決するには，次の三つの課題を解決する必要がある．

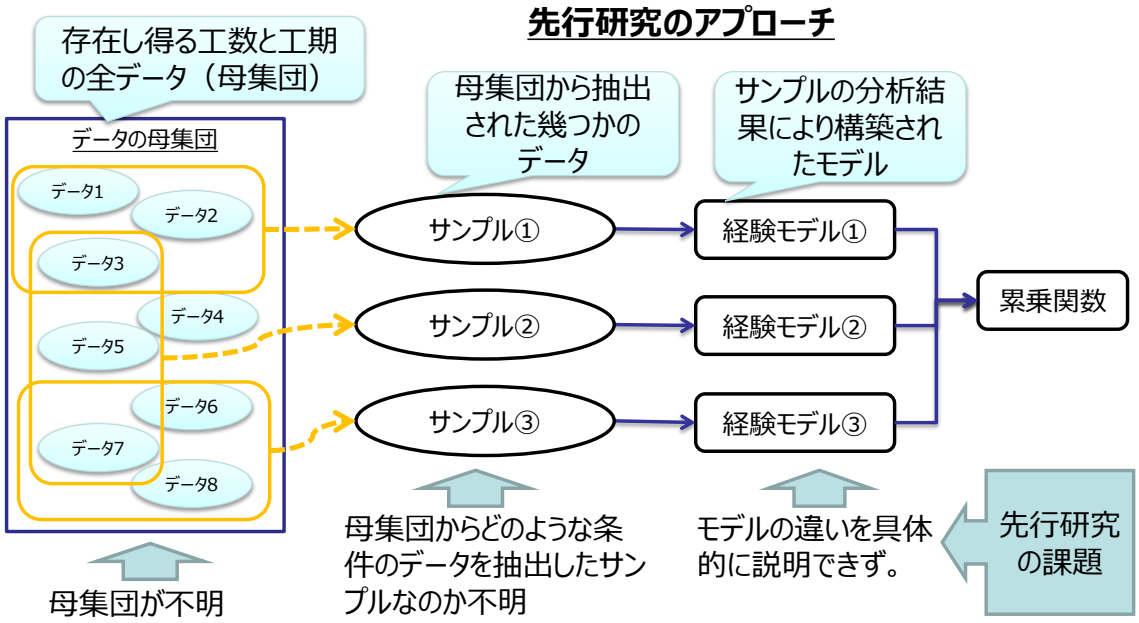


図 3-1 先行研究のアプローチと課題

Figure 3-1 Problem and Approach of Earlier Studies

課題 1 は，図 3-1 にも示したように，母集団データのデータ分布を理論的に明らかにすることである．経験モデルは，いくつかのサンプルデータを用いて構築されたが，そもそも，各モデル構築に用いられた各サンプルデータの全てを包含した母集団の分布状態・分布条件は分かっていない．どのような環境下のプロジェクトのデータが，母集団のどこに分布するのかが分からないのである．すなわち，各モデル構築に用いた異なるサンプルデータの条件の違いが明確ではない．従って，そのサンプルデータで構築されたモデルの乗数・指数が異なる理由も，説明することは難しい．

次に課題 2 は，理論モデルからメタモデル（$y=ax^b$）を導出することである．すなわち，何故，工数－工期の関係式が $y=ax^b$ の形式で，多くのモデルが一致するのか，説明が出来ていないのである[1]．そのため，工数－工期とその決定要因とその関係を，理論的に解明する必要がある．

最後に課題 3 は，理論モデルから経験モデルを導出することである．Putnam モデル[11]では，1 ヶ月かけて行う 100 人月の仕事を，2 ヶ月かけて行った場合，理論的には，6.25 人月で済んでしまう．他のモデルに比べて極端であるとの指摘がある[1]．また，Putnam モデルは経験論に対する説明もないとの指摘もある[32]．

以上を踏まえて本論文では，課題 1 より，工数と工期の母集団データを理論的に明らかにする．次に課題 2 より，その母集団データから，経験モデルが何故累乗関数（$y=ax^b$）で表されるのか，また，その乗数，指数が何故様々な値を取るのか，理論的に明らかにする．そして課題 3 として，導出した理論モデルと経験モデルを比較検証することで，その整合性を確認することを目的とする．

理論モデルはプロジェクト内でやり取りされる設計情報の写像経路と，その写像の難しさを変数としたモデルを用いる．以下に何故そのようなモデルを用いたか理由を述べる．この理論モデルは写像経路やその写像の難しさに着目したモデルであり，他の理論モデルのようにプロセスの組立て方で工数と工期を試算する方法とは異なる．前述の先行研究として挙げた理論モデル（DSM[12]など）では，特にプロセスに着目したモデルが多く，プロダクトの構造や要求仕様が工数や工期に及ぼす影響を定性的に議論していても数学的に整理するには至っていない．しかし，経験モデルでも COCOMOⅡ[2]などでは，プロダクトや要求仕様も考慮した説明がされている．従って，経験モデルと比較検証する上で，その精度を向上するためには，プロセスだけでなく，プロダクトや要求仕様を要素として考慮した理論モデルを用いることが必要と考えたことが理由である．

このような理論モデルを用いて，本論文の結果，経験モデルで示された累乗関数の乗数は写像経路とその写像の難しさに依存した関数で規定でき，またその指数は写像経路数のみに依存した関数で規定できることを示す．そして，これらの関数を元に理論的に導出した近似式と，実際のプロジェクトデータとを比較し，本理論モデルが十分に実際のプロジェクトを説明していることを実証検証する．

本論文の構成は次の通りである．まず 3-2 章で写像経路に着目したプロジェクトモデルから，工数－工期の決定要因とその関係を明らかにする．次に 3-3 章で理論モデルを構築する．3-3-1 節では，3-2 章で導かれた関係を元に，プロジェクトとして生じ得る工数－工期のデータの母集団を示す．全データの母集団を元に，工数－工期のメタモデルとして，近似曲線 $y=ax^b$ を導出する．そして，近似曲線が $y=ax^b$ の形式となる理由を考察する．更に 3-3-2 節では，3-2 章で示した工数－工期の決定要因が近似曲線の各係数（a,b）へどのような影響を及ぼすのか明らかにし，理論モデルを構築する．その後，3-4 章において，実際のプロジェクトデータと，3-3 章までで導き出した理論モデルによる近似式（累乗関数）を比較し，本論文で示した理論モデルによって，経験モデルを説明できることを検証する．最後に，3-5 章でまとめる．

3-2工数－工期の基礎理論

本章ではまず，先行研究における工数－工期の捉え方を説明し，次に本論文における工数－工期の捉え方を説明する．

3-2-1工数－工期の決定要因とその関係

3-2-1-1先行研究における工数と工期の捉え方

先行研究[9]において，多くのコスト要因が紹介されている．これらのコスト要因は，工数や工期に影響し，数学的に捉えるならば，工数や工期は多変数関数(f_c, f_t)とみなすことができる．ここで，x_iはコスト要因とする．

$$工数=f_c(x_1,x_3,\cdots x_i,\cdots) \tag{3-1}$$

$$工期=f_t(x_1,x_2,\cdots x_j,\cdots) \tag{3-2}$$

Boehm[33]がバリューチェーンで示したように，各要素が様々に影響して，“コスト”を構成している．そのため，工数や工期とその決定要因の関係を理解することは難く，そのコントロールも難しいと考えられている．

3-2-1-2本論文における工数と工期の捉え方・考え方

本論文では，工数－工期を以下のように捉え，その関係を検証する．まず，情報写像モデル[34]では，工数及び，生産性(1 人あたりの平均作業日数)が“複雑性”という変数と相関があることを示した（詳細は下記 3-2-1-3 で述べる）．

$$工数 \propto 複雑性\times要員数 \tag{3-3}$$

$$生産性（1人あたりの平均作業日数）\propto 複雑性 \tag{3-4}$$

$$複雑性=写像の難しさの平均\times全写像経路数 \tag{3-5}$$

なお，本論文では生産性を狭義にとらえ，式(3-4)の通り，1 人あたりの平均作業日数としている[34]．

上記をもとに，本論文では生産性（1 人あたりの平均作業日数）と工期の関係を明らかにし，工期が，工数と複雑性の関数 F となっていることを示す．

$$工期=F（工数，複雑性） \tag{3-6}$$

３-２-１-３ ソフトウェア開発プロジェクトモデル[34]

本論文で用いるプロジェクトモデル，写像，複雑性などは，2 章の通り，筆者の過去の研究[34]に基づく．本小節はそれをまとめる．まず，プロジェクトモデルは情報写像モデルと呼ばれ，公理的設計論[35]に基づく（図 3-2）．

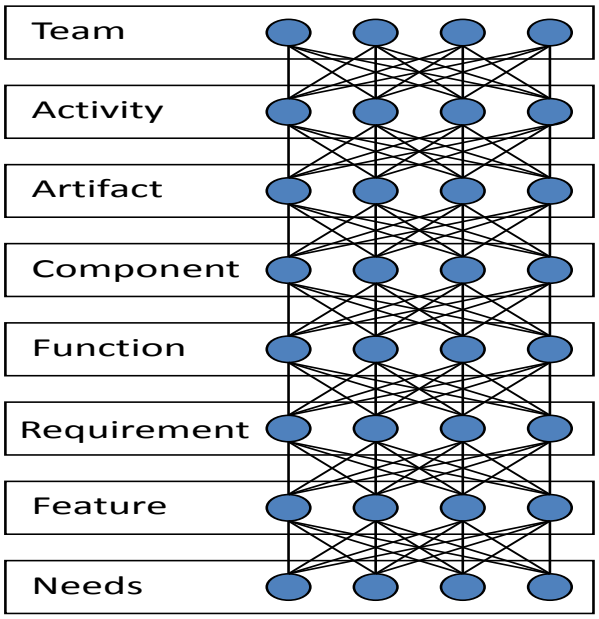


図 3-2 本論文で用いるプロジェクトモデル

Figure 3-2 Software Development Project Model

これによりソフトウェア開発プロジェクト内での設計情報の写像過程を表現する．本モデルでは，プロジェクトを構成する要素として，Team, Activity, Artifact, Component, Function，Requirement，Feature，Needs を抽出する（図 3-2 の丸印）． 実際にモデリングするにあたっては，例えば，Team とは個々の開発者とするケースもあれば，要求管理チーム，アーキテクチャー設計チーム等，プロジェクト内のチームとすることも可能である．Activity は開発で行われる作業である．Artifact とは，自然言語等で記述された設計書やソースコード等を指す．Component は個々のパッケージ，または更に詳細な粒度でモデリングするならば，Java クラス等を指す．その場合，Function は個々のメソッド等が該当する．Requirement は要求仕様，Feature は基本要件，Needs はニーズである．

また，これら要素間を結んだ線（図 3-2 の実線）は，設計情報の写像経路を示す．写

像とは，例えば自動車開発においては，自動車という 3 次元で表される実体と，2 次元で表現された設計書が存在する．どちらも同じ自動車を対象としているが，一方は 2 次元で，また一方は 3 次元で自動車という像を写し出している．このような関係を写像と言う．本モデルは，その各像を構成する要素をトレースし，それを線で結ぶことで，写像経路として示している．

３-２-１-４ ソフトウェア開発における写像過程

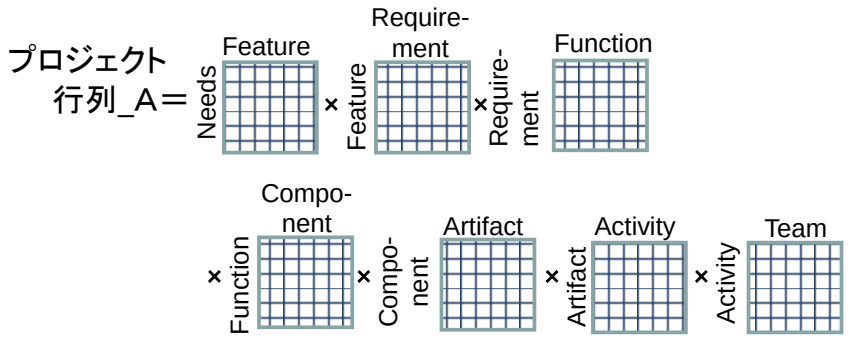
次に，このような写像という考え方によって，開発作業がどのように解釈されるのか説明する（詳細は，2 章を参照）．

例えば，複数の設計書(Artifact)によって，ひとつの部品(Component)が設計されている場合，Artifact と Component の写像関係は n 対 1 の関係になる．この時，ある開発担当(Team)は，ある設計作業(Activity)を通して，他開発者(Team)の設計書(Artifact)を参照し，自担当の設計書(Artifact)との整合を取ることで，ひとつの部品(Component)を正しく作ることが出来る．万一，ある設計書にひとつのミスが発見されれば，その設計書の作成作業の手戻りは勿論，多くの関連作業の手戻りにもつながり，生産性低下の要因となる．更に，写像関係が 1 対 1 であったとしても，例えば設計書が独特な表記によって作成されていたのであれば，それを部品に置き換えることは難しい．独自表記を読み解き，正しく理解し，誤りなく部品を作成することが必要となる．そのため，ある既知のルールに基づいた設計書を元にした場合に比べ，読み解き，正しく理解するために作業時間を要し，生産性は低下する．このようにプロジェクトを構成する要素の特性によっても，設計書から実体へ設計情報を写像することの難しさも変わる．すなわち，写像経路数と同様に，写像の難しさによっても生産性は変化する．

以上のような考え方から，設計情報の写像過程において，写像経路が少ない程，作業の生産性は高く，また，プロジェクトを構成する各要素の写像が容易な程，生産性が高い．即ち，要素間の相互依存は冗長性が低く，且つ要素自体は写像が容易な特性であることが望ましい．一方，式(3-4)より，生産性が高まることは複雑性が低くなることと同意である．従って，次節では，この複雑性の定義について説明する．

３-２-１-５ 情報写像モデルの数理的表現

先行研究[34]では，この情報写像モデルで示された要素間の関係を 7 つの行列の積で表現する数学的モデルを示した(式(3-7))．



(3-7)

このプロジェクト行列 A により写像経路数，写像の難しさを算出する．各要素の持つ写像の難しさを 0 以上 2 未満の値で評価し（1.0 を基準とする．また 0 は要素間の関係が無い状態を示す)，その相乗平均値を成分値とした各行列の積を行列 A としたとき，プロジェクトの写像の難しさを式(3-8)で定義した（a_{ij}は行列 A の成分値）．

$$写像の難しさ = |A| = \Sigma a_{ij} \tag{3-8}$$

また，ひとつの写像経路は，図 3-2 に示したように，Team から Needs までの各要素を縦につなげた 1 列である．隣の列の要素と相互依存すること無く，Team，Activity，Artifact という様に，上下に 1 つずつ相互依存する．このような写像経路の数を並行経路数と呼び，

$$並行経路数 = 行列A\ の次数 \tag{3-9}$$

と表した．それ以外の写像経路は，

$$冗長な経路数 = |A' - E| = \Sigma\ |\ a'_{ij} - e_{ij}\ | \tag{3-10}$$

（e_{ij}は単位行列 E の成分値）

として定義した．但し，行列 A' は行列 A の成分値を 0,又は 1（要素間の関連があれば 1，関連がなければ 0）とした行列であり，E は単位行列である．a'_{ij}は行列 A'の成分値を示す．

これらの写像経路数と写像の難しさの平均という 2 つの変数により，プロジェクトの複雑性を以下の方程式で定義した．但し，式(3-11)の写像の難しさの平均とは，式(3-8)を式(3-9)(3-10)の合計本数で割った値である．

$$複雑性 = 写像の難しさの平均 \times 並行経路数 \times \left(1 + \frac{冗長な経路数}{並行経路数}\right) \quad (3\text{-}11)$$

$$= \sum_i^{全写像経路数} 写像の難しさ(i) \quad (3\text{-}12)$$

なお，式(3-5)は式(3-11)を変形した結果である．そして，先行研究[34]では，この複雑性をもとに式(3-3)(3-4)が成立することを実証的に示している．

なお，この情報写像モデルは正方行列を前提としている．しかし，必ずしも各要素が全て同じ数となるとは限らない．従って，以下の方法で正方行列化することが必要である．要素数が不一致の場合は，ダミーの行または列を作成し，全ての要素数を合わせ，正方行列を作る．その際，そのダミーの行や列の成分値は 0 とする．0 を設定する理由は，あくまでもダミーの行や列であり，実際には要素間の関係が生じている訳ではないためである．写像経路数や写像の難しさを算出する上でも，成分値を 0 とすることで正しい値が求められる[34].

3-2-2 本論文における工数と工期の関係式

工数と工期の関係を前節で説明した先行研究を踏まえて整理する．まず一般的に工数と工期，要員数の関係は以下のように表される．なお,プロジェクトで行われる作業項目を Work Breakdown Structure（WBS）[36]として記述する．

$$工数 = 工期 \times 要員数$$
$$= \Sigma\{WBS(i)の作業期間 \times WBS(i)の要員数\} \quad (3\text{-}13)$$

ここで，WBS(i)を写像経路 i の写像作業と捉えると，式(3-13)は次のように変形できる．

$$工数 = \Sigma\{写像経路(i)の写像期間 \times 写像経路(i)に関わる要員数\} \quad (3\text{-}14)$$

すなわち，プロジェクトの工数は，ある写像経路 1 本の写像作業を担うサブプロジェクトの工数の総和となる．写像経路 i を担うサブプロジェクトでは，式(3-12)より，その複雑性は，写像経路 i の写像の難しさで決まる．更に，式(3-4)より，その複雑性は，1 人あたりの写像経路 i の平均作業日数, すなわち, 写像経路 i の写像期間に比例する．従って，式(3-14)は次のような比例関係で表せる．

$$工数 \propto \Sigma\{写像経路(i)の複雑性 \times 写像経路(i)に関わる要員数\} \quad (3\text{-}15)$$

前述の通り，写像経路 i の複雑性は，式(3-12)より，写像経路(i)の写像の難しさに等しいので．式(3-15)は次のようにも書き換えられる．

工数∝Σ{写像経路(i)の写像の難しさ　×　写像経路(i)に関わる要員数}　　(3-16)

写像の難しさを 1 とすれば，ある写像経路に関わる要員数の合計が工数に比例することとなる．

3-2-3 写像経路数・要員数・工期の組合せの整理

前節を元に，プロジェクトで存在し得る写像経路数，要員数及び工期の組合せを洗い出す．

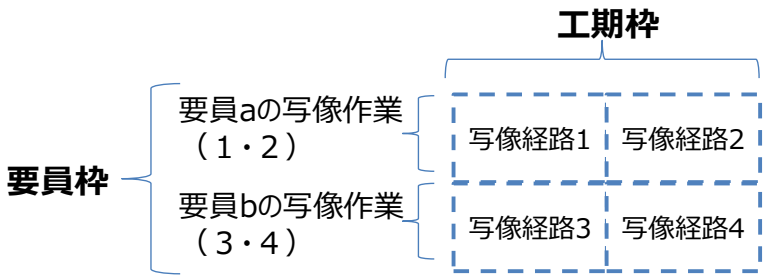


図 3-3　要員数・工期・写像経路数の図形的解釈

Figure 3-3 Diagram Analysis for Relation between Mapping Paths and Personnel, Development Schedule

まず，式(3-16)を図 3-3 のような長方形のモデルで考える．図 3-3 では，プロジェクトの各要員の割当先となる要員枠を短辺に取り，各写像作業の工期枠（写像の難しさの平均値 k を 1 ヶ月/人とすれば，1 つの工期枠は 1 ヶ月．但し，以降では，写像の難しさは，式(4)における比例係数を含めたものとした．）を長辺に取っている．図 3-3 では，写像経路 4 本を取り上げており，要員数 2 人，工期 2 か月，すなわち工数は 4 人月プロジェクトが完遂できることを示している．写像経路 1 と 2 は，写像の難しさ 1 ヶ月/人で要員 a が 1 人で作業する．また，写像経路 3 と 4 は，写像の難しさ 1 ヶ月/人で要員 b が 1 人で作業する．式(3-16)によれば，これらの合計がプロジェクトの全工数となる．

図 3-4　写像経路 4 本の時の要員と工期の割当て方法

Figure 3-4 In case of 4 Mapping Paths, Variations of Developers and Development Time

更に写像経路 4 本の場合では，要員数・工期について，図 3-4 のようなバリエーションが考えられる（図 3-4 の(3)は，図 3-3 と同じ）．以下で，そのバリエーションを説明する．

図 3-4 の(1)と(5)は，それぞれ要員数 1 人で工期 4 ヶ月，要員数 4 人で工期 1 ヶ月，すなわち工数は 4 人月というケースを示している．図 3-4 の(2)は，要員数 2 名で工期 3 ヶ月というケースを示す．このケースでは，要員の 1 人が 3 本の写像作業を行ってしまえば，他の 1 人は 1 本の写像作業をするのみでプロジェクトは完遂する（工数は 4 人月）．また別に，1 本の写像作業しか受け持っていない 1 人の要員に，もう 1 人の写像作業に加わってもらい，協働で写像作業をすることも考えられる．従って，写像経路 4 本を要員数 2 人で工期 3 ヶ月対処する場合のプロジェクトでは，工数が 4 人月，5 人月，6 人月となるケースが考えられる．上記は，要員 3 人で工数 2 ヶ月（図 3-4 の(4)）も同様である．

以上のように，ある写像経路数に対して，想定される全ての要員数・工期を一覧表に整理する．そして，その写像経路の枠数を数えることで，工数を求める．このような手順で，写像経路数，工期，工数の全ての組合せを洗い出し，プロジェクトの母集団データを整理する（その結果は，3-3-1 節でグラフ化して示す）．なお，上記では k＝1 を前提とした．k≠1 の場合は，式(3-16)により工期が k 倍（図形の長辺が k 倍）に伸びるが，長方形の枠数の考え方は変わらない．

３-３理論モデルの構築

前章の基礎理論を元に，本章では工数工期の母集団データがどのような分布になるのかを明らかにする．それを元に，近似式（メタモデル）を導出する．そして，導出した近似式の累乗関数の乗数と指数の特性を説明する．

３-３-１工数－工期の母集団データとその近似曲線

３-３-１-１母集団データとメタモデル

3-2-3 節で示した方法により，写像経路 1 本～10 本，15 本，20 本（写像の難しさの平均は 1）で取り得る工数と工期を洗い出した．その結果，母集団は図 3-5 の通りとなった．

また，その近似曲線は累乗関数，$工期=2.0009\times 工数^{0.3695}$ となった．相関は弱いが，次節 3-3-1-2 で示すように母集団データの分布特性上，その近似曲線の関数の型（累乗関数，対数関数等），すなわちメタモデルは，累乗関数となる．

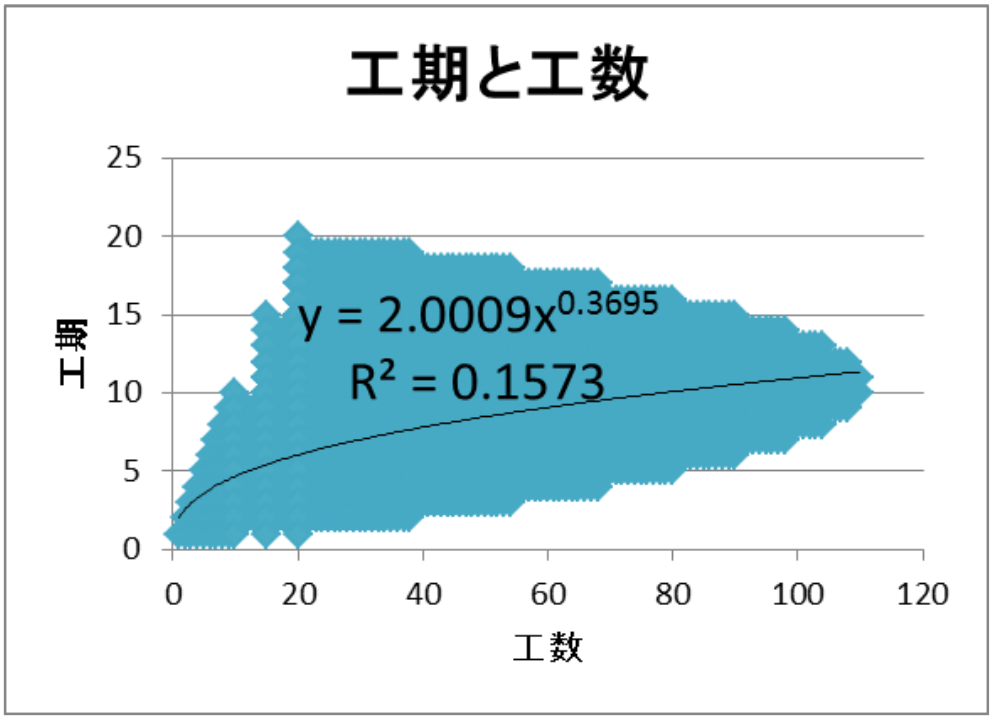


図 3-5　工数－工期の母集団データと近似曲線

Figure 3-5 Population of Man-Month and Development Time Data along with its Fitted Curve

なお，本グラフは写像経路数が前述の通り最大 20 本のケースである．更に写像経路数が多いプロジェクトを母集団データに組み入れた場合は，半楕円形のような図形で示されたデータ分布は膨れていく（詳細は 3-3-2 参照）．その結果，ある工数で取り得る工期の最大値も延びていく．例えば，失敗プロジェクトの多くは，要求仕様の曖昧さが指摘され，その整理や設計のために開発要員が増員される．これは図 3-2 に示したプロジェクトモデルで考えると，Requirement や Team の要素数が，プロジェクト開始当初に比較して，増加している状態である．すなわち，写像経路数は増大し，また曖昧な Requirements 等のため，写像の難しさの平均も増大している．このような状況下では，上記で述べたように，図 3-5 で示されたデータ分布も工数は増大する方向へ，そして工期も増大する方向へ膨らんでいくこととなる．その結果，近似曲線の形状も変化していく．その変化の特性については以降 3-3-2 節で説明する．

3-3-1-2 累乗関数となる要因

各工数で取り得る開発期間のデータ件数の分布傾向を示す．その傾向から，累乗関数となる理由を説明する．

図 3-6 は，図 3-5 を輪切りにした状態（各工数単位に，縦軸は開発期間，横軸はデータ数としたグラフ）を示した．図 3-4 で示したように，例えば工期 2 ヶ月で工数 4 人月という組合せは 2 通りある．このように図 3-5 の各工数ー工期のプロットは複数存在するものもある．図 3-6 のデータ件数はそれを示し，そのデータ分布の傾向は以下のようにまとめられる．なお，各工数別にデータ分布状況を示しており，各グラフの横軸はデータ件数を示しており，左側にいくほど件数が多くなることを示している．

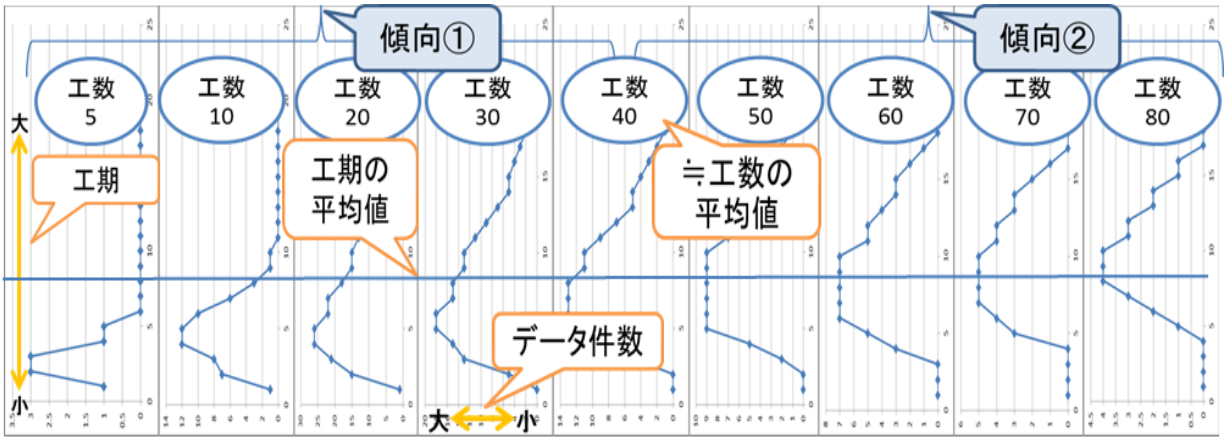


図 3-6　データの分布傾向

Figure 3-6 Distribution State of Data from Figure 3-4

傾向①として，工数が 5 から平均値の 39.7 程度まででは，開発期間の平均値（8.3）よりも小さい値をピークにデータが分布しており，平均値との乖離が大きい．傾向②として，工数の平均値（39.7）を超えたあたりから，ほぼ平均値程度の値をピークとしたデータ分布をしており，平均値との乖離が小さい．このため，各データと平均値との差分に着目する最小 2 乗法により，母集団の近似曲線を求めると，工数―工期の関係式は，指数が 1 以下の累乗関数となる．なお，工数=1 で工期=0 ではないため，対数関数は近似式とはなりえない．

3-3-2 メタモデルから理論モデルの導出

次に，メタモデルの性質について述べ，理論モデルを導出する．近似曲線は，プロジェクトの複雑性（写像経路数，写像の難しさの平均）を変化させると，近似曲線も変化する．この性質により，各経験モデルも説明できる．
本節では，工期が下記の式(3-17)で表される理論モデルとなることを示す．

$$\begin{aligned} 工期 &= f（写像経路数，写像の難しさの平均）\times 工数^{\,g（写像経路数）} \\ &= f（複雑性）\times 工数^{\,g（写像経路数）} \end{aligned} \qquad (3\text{-}17)$$

なお，上記理論モデルの各係数の具体的な決め方については，3-4-2 節で述べる．

3-3-2-1 写像経路数と乗数・指数の関係

本節では，写像経路数の変化が乗数・指数にどのように影響し，式(3-17)のような関係式となるのか説明する．まず，写像経路数を変えた際に，3-3-1 節で示した近似曲線がどのように変化するのか調べた．

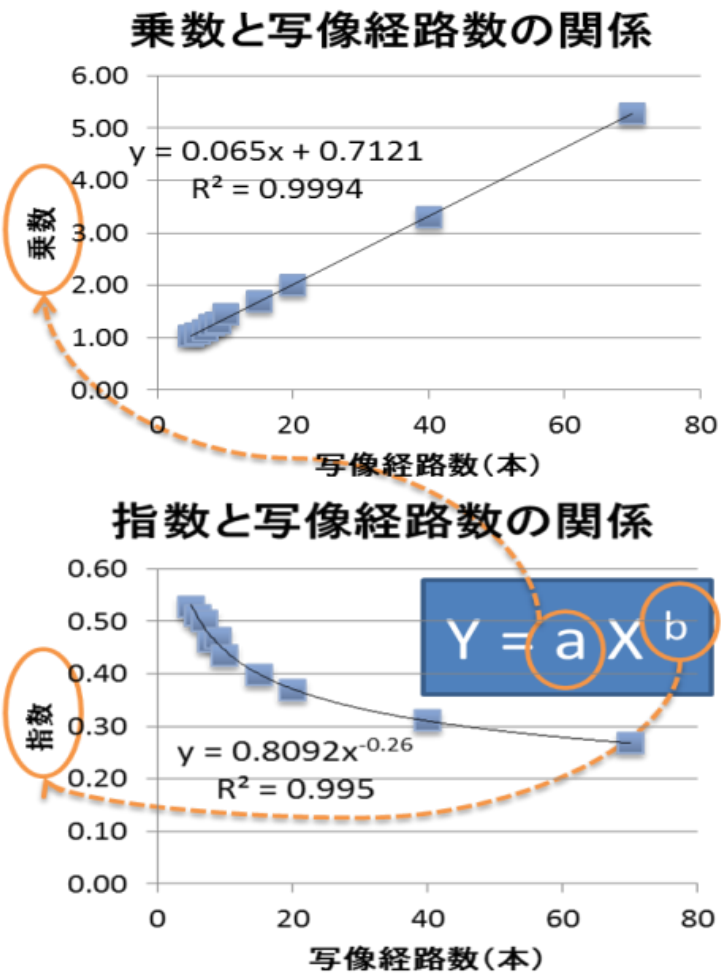


図 3-7 写像経路数と乗数・指数の関係

Figure 3-7 Relations between Number of Mapping Paths and Multiplier, Exponent

図 3-7 の上図のように，近似曲線の乗数と写像経路数の関係は，写像経路数が増えるに従い，乗数の値も増加する比例関係にある．その関係式は，

$$Y=0.065X+0.7121 \tag{3-18}$$

で表される．その R^2 は 0.9994 であり，高い相関関係にあることがわかる．また，近似式の指数は，図 3-7 の下図のように写像経路数が増加するに従って，減少する傾向があり，

$Y=0.8092X^{0.26}$ (3-19)

で表される．その R^2 は 0.995 であり，高い相関関係にあることがわかる．

３-３-２-２写像経路数による乗数・指数の変化要因

本節では，写像経路数の増減により，何故，前節のような変化をするのか説明する．図 3-8 は，前述の図 3-5 を拡大したものである．工数を横軸に，工期（開発期間）を縦軸にとっている．グラフ上の四角い点はプロジェクトで取り得る工数―工期のデータである．データを結んだ楕円の曲線はプロジェクトの写像経路数を示している．
すなわち，4 本の写像経路を持ったプロジェクトでは，工数が 4 人月で工期が 4 ヶ月，または工数が 6 人月で工期が 3 ヶ月，工数が 6 人月で工期が 2 ヶ月，工数が 4 人月で工期が 1 ヶ月というプロジェクトが存在し得ることを示している．但し，この時，写像の難しさの平均は 1 ヶ月／人である．

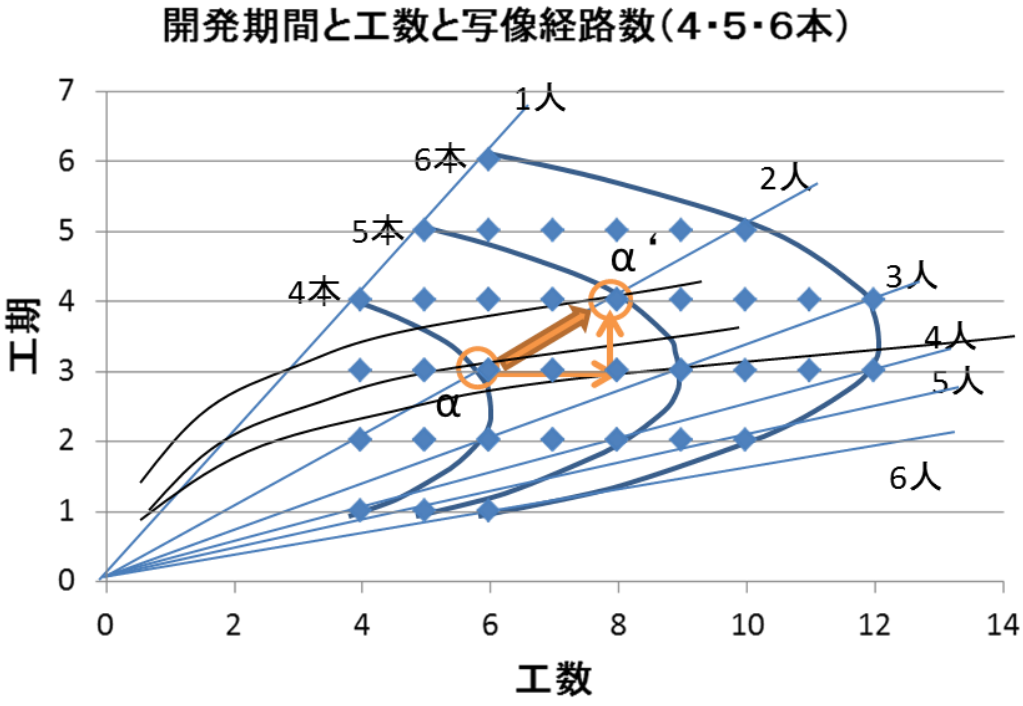


図 3-8　写像経路数と工数－工期の関係（図 3-5 の拡大図）

Figure 3-8 Relation between Mapping Paths and Man-Months, Development Schedule (Magnified Figure 3-5)

また，もうひとつデータを結んだ直線は，要員数を表している．すなわち工数 4 人月で工期 4 ヶ月ならば要員数は 1 名，また工数 5 人月で工期 5 ヵ月でも同様に要員数 1 名のプロジェクトである．

このようにグラフを見ると，例えば，工期 3 ヶ月・工数 6 人月のプロジェクト(α)に対し，経路を 1 本追加する．その際，追加した経路の写像作業分，工数が膨らむ．しかし，要員数は変わらない．そのため，α'にデータは移行する．その際，経路数が増加した場合の X 軸に対する変化を説明する．経路数が増加することによって，処理すべき写像経路数が増えたために工数が追加で必要となる．そのため，近似曲線は X 軸に対して伸びることとなる．即ち，近似曲線の指数が，小さい値に変化する．次に，経路数が増加した場合の Y 軸に対する変化を説明する．経路数が増加することにより，写像処理を行うための工期が必要となる．従って，元の経路数の時に比べて工期が延長する．即ち，近似曲線の乗数が，大きい値に変化する．

3-3-2-3 写像の難しさと乗数・指数の関係

本節では，当初 1 に固定していた写像の難しさの平均を変化させた場合に，近似曲線の乗数・指数の変化を調べた．

図 3-9 より，写像の難しさの平均が増すに従って，乗数は増加するが，指数は一定値を取ることが分かった．写像の難しさの平均に対する乗数の変化は

$$Y=2.0009\times X^{0.6305} \tag{3-20}$$

で表される．その時の R^2 は 1 であり，非常に高い相関関係にあることが分かる．
また，写像の難しさの平均に対する指数の変化は

$$Y=0.3695 \tag{3-21}$$

で表される．R^2 は 0 であり，写像の難しさの平均とは全く相関しないことが分かった．但し，本サンプルでは写像経路数が 20 本のプロジェクトを元にしているため，この値となっている．経路数を変えれば，式(3-21)の定数値も変化する．詳細は次節にて説明する．

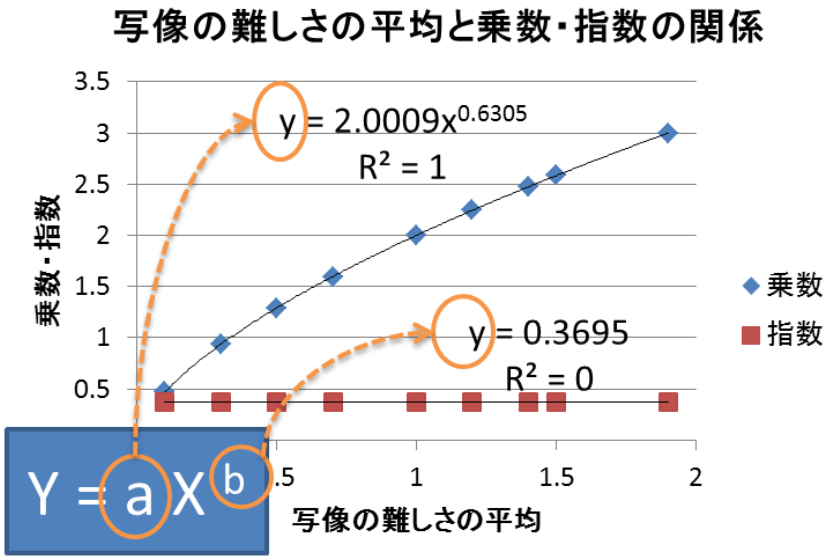


図 3-9　写像の難しさと乗数・指数の関係

Figure 3-9　Relation between Difficulty of Mapping Paths and Multiplier, Exponent

３-３-２-４写像の難しさによる乗数・指数の変化要因

本節では，写像の難しさの平均の変化が乗数・指数にどのように影響するのか説明する．

写像経路数が変わらないため，プロジェクトモデルの構造も変わらない.しかし，写像の難しさの平均値が k 倍になれば，工数と工期は k 倍される．従って，データ分布状態は変化しないため，指数は変わらず，乗数のみ変化する．

x 人月の工数で y ヶ月の写像作業（要員数 n 人）を行ったとき，乗数を a，指数を b とすると

$$y=a\times x^{b} \tag{3-22}$$

k 倍された工数（$x\cdot k$ 人月）の作業では，工期が $y\cdot k$ ヵ月となる（プロジェクト構造は変化しないので，要員数は n 人のまま）．

$$y\cdot k=a'\times (x\cdot k)^{b} \tag{3-23}$$

上記式(3-23)を(3-22)で割り，式を整理することにより，

$$a'=a\times k^{1-b} \tag{3-24}$$

すなわち，20 本の経路数における乗数と指数を元の定数，元の指数とすれば，

$$\text{乗数}=\text{元の乗数}\times\text{写像の難しさの平均}^{(1-\text{元の指数})} \tag{3-25}$$

として表される．

3-3-3 モデル粒度の規格化（補正方法）

前節までで説明した理論の正しさを検証するために，実際のプロジェクトデータと比較検証を行う．しかし，実際のプロジェクトデータを用いるためには，1 点留意すべき点がある．モデル化する際，それを構成する要素の粒度は規定されていない．従って，同じプロジェクトを異なる粒度でモデル化することもある．その場合，同じプロジェクトでも写像経路数や写像の難しさが異なってしまうため，それらを補正し，規格化する必要がある．

例えば，同じプロジェクトでも，以下の 2 つのようなモデル化が可能である．ひとつは，Team，Needs をそれぞれ 1 つの要素で，写像経路を 1 本でモデル化する．また別に Team，Needs をそれぞれ 2 つの要素で，写像経路で各要素を全て繋いだ 4 本の写像経路でモデル化することもできる．もし，この 2 つのモデルが同一のプロジェクトであるならば，工期は等しくなければならない．そこで，本論文では経路数が 1 本，すなわち Team，Activity など各カテゴリの要素を 1 つのみのモデルを基準とする．規格化前のモデルの経路数を，この基準に換算して規格化後の経路数を算出する．以下がその算出方法である．

上記の通り，経路数 1 本の基準モデルは，その要素の関係を表すプロジェクト行列(式(3-7))の次数が 1 次の行列として表現できる．その時，規格化前のモデルの次数 n'，また，その写像経路数 m' とすると，規格化前のモデルのプロジェクト行列は n'次の行列となり，Team と Needs の組合せは，基準モデルの $n'\times n'$倍となる．すなわち，$n'\times n'$ 倍の細かな粒度でモデルが構成されている．従って，以下のような比例関係が成り立つ

$$\begin{aligned}&1\text{ 本}:m'\text{本}=1\text{ 次行列の成分数}:n'\text{ 次行列の成分数}\\&\qquad=1\times 1:n'\times n'\end{aligned} \tag{3-26}$$

式(3-26)を変形し，以下の換算式を導出した．

$$m' \div (n' \times n') = 1 \tag{3-27}$$

写像の難しさの平均についても同様に考え，以下の関係式を導出した．（k は規格化後，k'は規格化前の写像の難しさの平均）

$$k = k' \times n' \times n' \tag{3-28}$$

上記の 2 つの式で規格化された k，m に換算・補正する．

3-4 理論検証

前章で理論的に導いた複雑性と乗数・指数の関係式を，実際のプロジェクトのデータに当てはめ，理論的に近似式を導出する．そして，実際のプロジェクトデータと比較検証を行い，本論文で説明した理論の整合性を確認する．

3-4-1 プロジェクトの概要

検証にあたっては，ゲームソフトウェア開発プロジェクトの開発データを用いた．対象は 21 プロジェクトであり，それらの工期・工数・写像経路数・写像の難しさの平均値は表 3－2 の通りである． 但し，Boehm[1]と同様に，1 ヶ月は 19 日として工数－工期を算出している．また，写像の難しさの平均値は以下のように算出した．
式(3-3)を変形すると，

$$工期 = \frac{工数}{要員数} \propto 複雑性 \tag{3-29}$$

となる．更に式(3-5)より，式(3-30)のように変形できる．

$$工期 \propto 写像の難しさの平均 \times 全写像経路数 \tag{3-30}$$

従って，写像の難しさの平均値は式(3-31)によって求まる．

$$写像の難しさの平均 \propto \frac{工期}{全写像経路数} \tag{3-31}$$

なお，表 3-2 は，写像の難しさの平均値を定数倍し，式(3-31)を比例式ではなく等式とした場合の値である．

表 3-2 実プロジェクトの検証用データ

Table 3-2 Data for Validation of Actual Software Development Projects

プロジェクト	工数（人月）	工期（月）①	全写像経路数②	写像の難しさの平均値（①÷②）
1	3.187	0.83	1566	5.28.E-04
2	3.761	0.97	1566	6.19.E-04
3	2.850	0.74	1566	4.74.E-04
4	3.334	0.86	1566	5.51.E-04
5	3.008	0.78	1566	4.99.E-04
6	2.950	0.77	1566	4.90.E-04
7	3.342	0.70	1794	3.92.E-04
8	2.618	0.68	1566	4.33.E-04
9	2.861	0.74	1566	4.71.E-04
10	2.782	0.72	1566	4.59.E-04
11	2.834	0.73	1566	4.67.E-04
12	3.155	0.81	1566	5.18.E-04
13	3.303	0.85	1566	5.42.E-04
14	3.111	0.68	1794	3.77.E-04
15	2.442	0.63	1566	4.02.E-04
16	2.605	0.67	1566	4.28.E-04
17	2.337	0.60	1566	3.85.E-04
18	2.863	0.73	1566	4.69.E-04
19	2.495	0.64	1566	4.10.E-04
20	2.758	0.71	1566	4.52.E-04
21	2.621	0.57	1794	3.17.E-04

3-4-2近似式の理論的導出

本節では近似式の理論モデルを策定する． はじめに，写像経路数の補正値を算出する．補正値は前述の式(3-27)より，以下の算出式から求める．なお，補正前の写像経路数・写像の難しさは，表 3-2 の全 21 プロジェクトの平均値を用いた．

m=全 21 プロジェクトの写像経路数平均値÷(行列の次数×行列の次数)
=1599÷（18×18） (3-32)

この補正値 m を用いると，乗数係数と写像経路数の関係式（式(3-18)(3-19)）により，乗数と指数は以下の通りとなる．

$a_0 = 0.065 \times m + 0.7121 = 1.0329$ (3-33)

$b = 0.8092 \times m^{-0.26} = 0.5343$ (3-34)

次に，写像経路数と同様に，写像の難しさの補正を式(3-28)により算出する．

k=写像の難しさの全 21 プロジェクトの平均値×(行列の次数×行列の次数)
=0.000461×18×18 (3-35)

この補正値 k を用いて，指数係数と写像の難しさの平均の関係式（式(3-24)）より，乗数を確定する．

$a = a_0 \times k^{1-b} = 0.4261$ (3-36)

となる．
以上から，工期と工数の関係式は以下のように定義される．

工期 $= 0.4261 \times$ 工数 $^{0.5343}$ (3-37)

3-4-3 工数－工期の理論値と実測値の比較検証結果

前述の通り，全プロジェクトの平均値から算出した工期と工数の関係式の理論曲線（式(3-37)）と実測値を比較する．図 3-10 の通り，実測値は，理論曲線上の近傍に現れている．以下で，この理論値と実測値の関係について考察する．

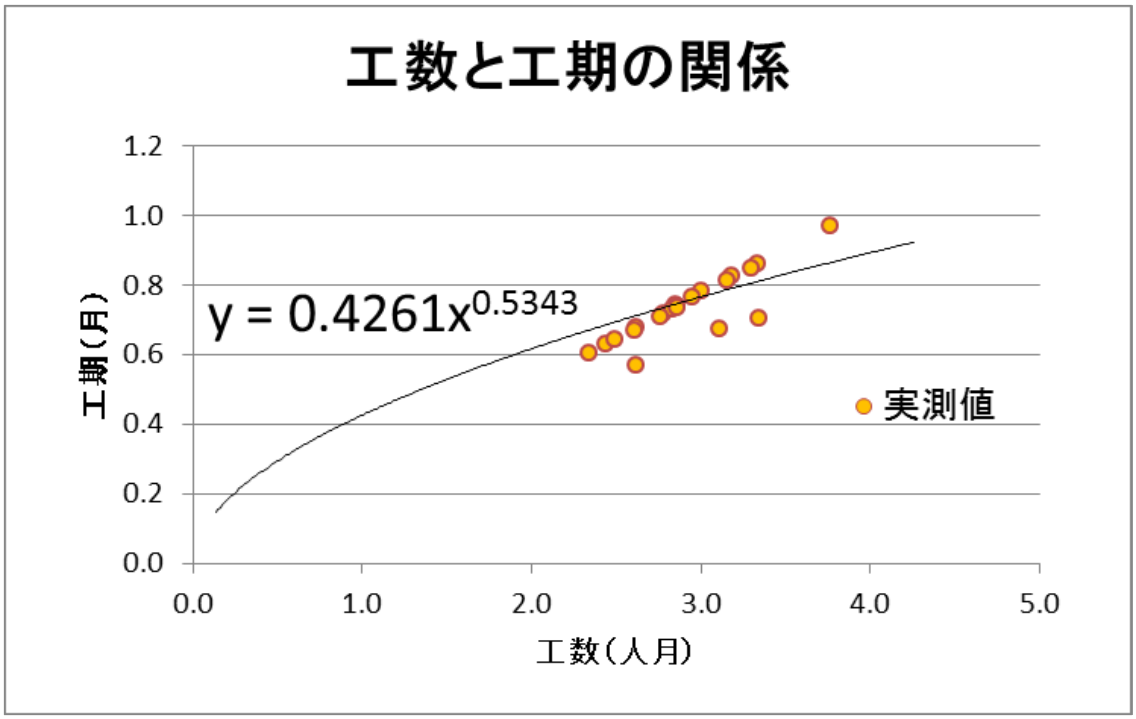


図 3-10 理論値と実測値の比較結果

Figure 3-10 Comparison of Theory and Measured Results

まず，経験モデルとして，実測値から線形近似（式(3-38)）を導出した．

$$工期 = 0.2343 \times 工数 + 0.0508 \quad (3\text{-}38)$$

式(3-38)と実測値の相関係数は 0.867 である．同様に，実測値から累乗近似（式(3-39)）を導出した．

$$工期 = 0.274 \times 工数^{0.9191} \quad (3\text{-}39)$$

式(3-39)と実測値の相関係数は 0.858 である．線形近似より多少相関が弱い．一方，理論的に求めた累乗近似（式(3-37)）では，相関係数が 0.865 となり，線形近似の場合に近い値となった．なお，p 値は，どのケースも 0.005 以下である．

以上のように，今回の実証検証では，線形近似の方が累乗近似よりも相関係数が高い結果となった．しかし，一般的に経験モデルが累乗関数となることは，表 3-1 でも示した通り，多数の研究で言われていることである．本論文で示した理論モデルは，経験モデルの累乗近似よりも高い相関を示し，且つ，本検証で用いた 21 プロジェクトの実測値とは高い相関を示した線形近似と比較しても遜色のない相関係数となっている．従って，本論文で示した理論モデルは経験モデルのメカニズムを明らかにしたと考える．

3-4-4 工期の実測値と理論値の相関と誤差

前節では，21 プロジェクトの平均値より理論モデルを求めた．本節では，21 プロジェクト個々の実測値と理論値の比較結果を示す．

表 3-2 で示した各プロジェクトの写像経路数，写像の難しさの平均により理論式を算出し，実測値の工数を代入することで工期-理論値を求めた．また，理論値で用いた工数に対応する実測した工期を工期-実測値とした．この 2 つの工期を Y 軸，X 軸に取り，グラフ(図 3-11)を作成した．すなわち，理論値が正確に実測値を再現しているほど，近似式の傾きは 1 に近い値となり，また y 切片も 0 に近いこととなる．図 3-11 のように本論文で構築したモデルは，傾きも 1 に近く，また y 切片も 0 に近い．従って，理論値は実測値を正確に再現できていることが分かる．実際，その相関係数は 0.982 である．また，理論値と実測値の誤差は 1 日以下であり，工期の実測に対し，理論値との誤差は「-0.40 日～0.7 日」とであった．以上から，本論文で構築した理論モデルは，個々の実測値をほぼ再現できるモデルであると考える．

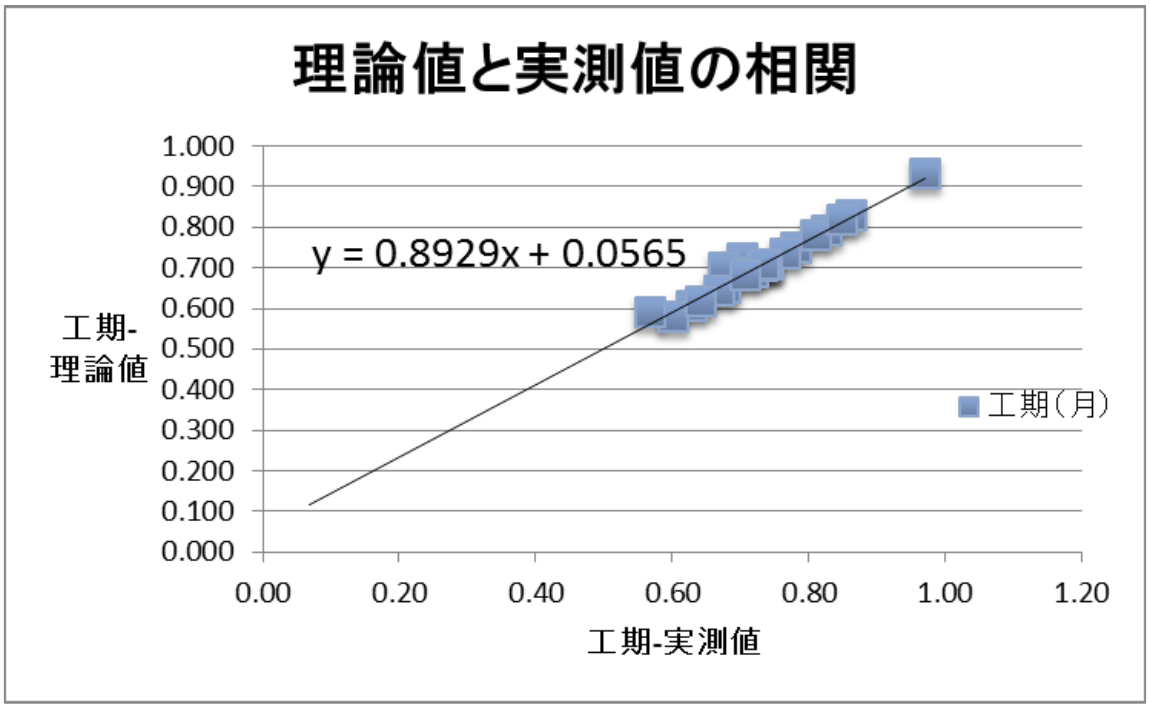


図 3-11 理論値と実測値の相関

Figure 3-11 Correlations between Theoretical Value and Observed Value

3-5まとめ

本論文では，工数と工期の経験モデルが何故累乗関数（$y=ax^b$）で表されるのか，そして，その乗数，指数が何故様々な値を取るのか，理論モデルを用いて明らかにした．その理論モデルはプロジェクト内でやり取りされる設計情報の写像経路と，その写像の難しさの平均を変数としたモデルを用いた．本論文の結果，乗数は写像経路とその写像の難しさの平均に依存した関数で規定され，指数は写像経路数のみに依存した関数で規定されることを示した．

そして，これらの関数を元に理論的に算出した近似式と，実際のプロジェクトデータとを比較し，本理論モデルが十分に実際のプロジェクトを再現していることを実証検証した．すなわち，写像経路数，写像の難しさの値が様々なプロジェクトを元に，工数－工期のモデルを構築したため，先行研究のモデルは様々なものとなっていることが分かった．そして同時に，以下に示す通り，先行研究の課題を解決した．まず，課題 1 として，経験モデルのデータ分布状態を理論モデルから説明することが出来ていなかった．本論文により，理論モデルから，発生し得るデータ分布（データの母集団）を明確化した．次に，課題 2 として，理論モデルからメタモデルを導出することが出来ていなかった．本論文により，データの母集団から近似曲線を求め，関係式 $y=ax^b$ を導出した．最後に，課題 3 として，理論モデルから経験モデルを導出することが出来ていなかった．本論文により，$y=ax^b$ の各係数値の決定要因は，写像経路数と写像の難しさであること示した．以上により，工数－工期の関係における課題を解決した．

次章では，本論文の結果を用いて，どのようにプロジェクトをマネジメントしていくのか説明する．

【4 章】

複雑性指標を用いた
プロジェクトマネジメントの展望

４．　複雑性指標を用いたプロジェクトマネジメントの展望

第 1 章で，プロジェクトの失敗（コストオーバー，納期遅延）の多くが，その複雑性に起因していると考えられていると述べた．それに対し，3 章までで，プロジェクトに関わる様々な要素（Needs から Team まで）の関連性を整理するモデルを紹介した．そして，そのモデルから算出される複雑性指標により，プロジェクトマネジメントの基本となる工数と工期の関係性を理論的に導いた．これにより，プロジェクトの失敗要因を解決する情報写像モデルと複雑性指標という技術とツールを手に入れることが出来た．本章では，手に入れた技術とツールを如何に使いこなしていくべきか，その展望について述べる．

4-1 既存のマネジメント技術の課題

まずは，旧来のマネジメント技術について，その状況を整理する．PERT/CPM 等，プロセス中心のマネジメント技術は第 1 章でも述べた通り，その限界を指摘されている[1]．指摘された課題は，①日常的に生じる手戻り管理が難しい．②はじめに計画したプロセス（作業順序性）の変更に弱い．③プロジェクト内で生じている問題が分かり難い．これら 3 つの問題が明確に指摘されている．この指摘に関連して，旧来のプロセス中心の開発については多くの指摘がなされている[2][3][4][5][6][7]．プロセス中心のマネジメント技術は，プロセスが一方向に進むことを前提として考えられ，その順序性によってスケジュールを組んでいる．そのため，手戻りや変更に弱いことは当然なことと考えられる．また作業の進捗状況をパーセンテージで捉え，進捗把握をすることが主流であろう．進捗が計画より進んでいるか否かは分かっても，例えば PERT/CPM を見るだけで進捗の障害となっている問題は何なのか判別することは難しい．

一方，複雑性指標を用いたマネジメントにおいては，これらの課題を次のように克服している．①写像経路の存在有無により，手戻り管理が可能である．②そもそも作業の順序性を規定していないため，作業順序の変更に対して影響がない．③写像経路数・写像の難しさの変化が，プロジェクトにおける問題発生箇所を浮き彫りにしてくれる．例えば，一旦完了した写像経路で再度写像を行うことになったとしても，モデルに再度線が引かれ，写像経路数が増加し，複雑性指標が増加することになる．複雑性指標が増加すれば，工数オーバーまたは工期遅延のリスクが生じていることとなり，プロジェクトとしてその対処策を講じる必要が出てくる．その際には，当然問題の所在を明らかにしなければならない．複雑性指標を用いたマネジメントでは，上記のとおり，写像経路が新たに追加された箇所で問題が生じているのである．このようにプロジェクト内で生じた問題箇所も明確に示されるのである．以上のように，設計情報の写像関係に着目したマネジメントは，プロセス中心のマネジメントの弱点を克服したと考えている．以降では，複雑性指標を用いたマネジメント方法について，その詳細を説明していく．

4-2 複雑性（写像経路数と写像の難しさ）の時間変化

前章まではプロジェクトの時間軸を考慮せず，ある時期でのスナップショットとして議論していた．本小節では写像経路数と写像の難しさの時間的変化について検討を行う．

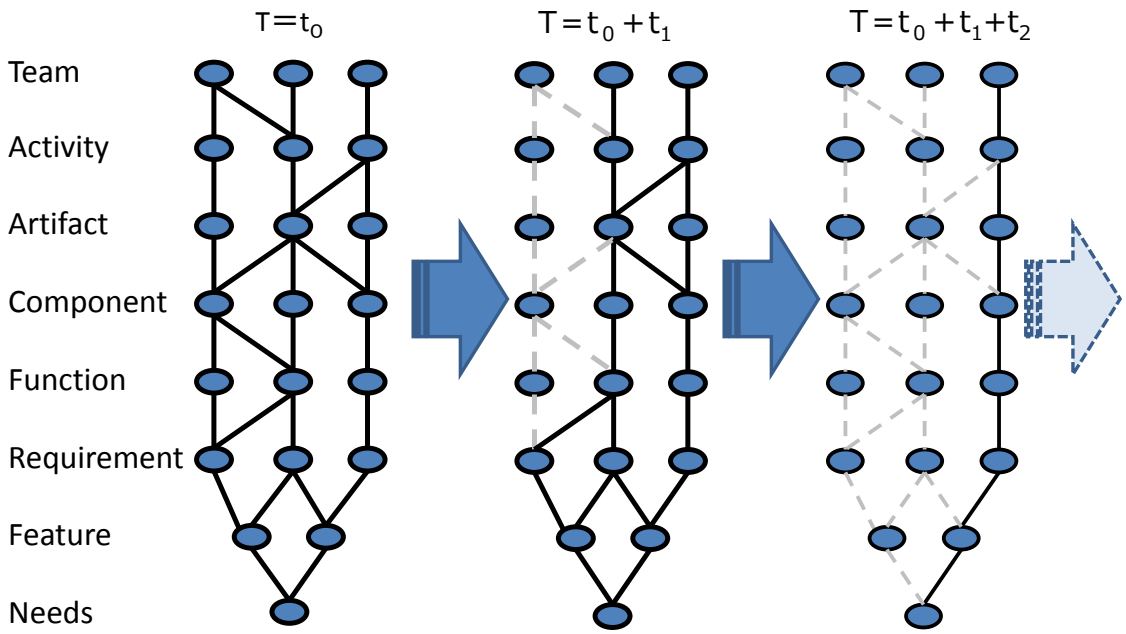


図 4-1 写像経路の時間変化

Figure 4-1 Alteration of Mapping Paths from Moment to Moment

図 4-1 に示したように，プロジェクトの作業が進むにつれ，写像が完了する．従って，作業が順調に進むのであれば，完了した写像経路は相互依存関係が無くなっていく．従って，写像経路数は減少していくはずである．また，時間が経つにつれ，例えば開発者の習熟度も向上し，写像の難しさも低減していくはずである．その結果，プロジェクトの複雑性はゼロに収束していくことが期待される（図 4-2）．一方，何か問題が生じているプロジェクトの場合は，例えば，手戻りによって一度完了した写像でも，再度写像し直すことも想定される．即ち，減少傾向にあった写像経路数が増加に転じる．その結果，複雑性が増大することが想定される．このように複雑性が増加傾向にあるか，減少傾向にあるかによって，プロジェクトが順調か否かの判断ができる．このように複雑性の傾向を元にマネジメントするため，手戻りもマネジメントできるし，また作業の順序性の変更にも柔軟に対応できる．

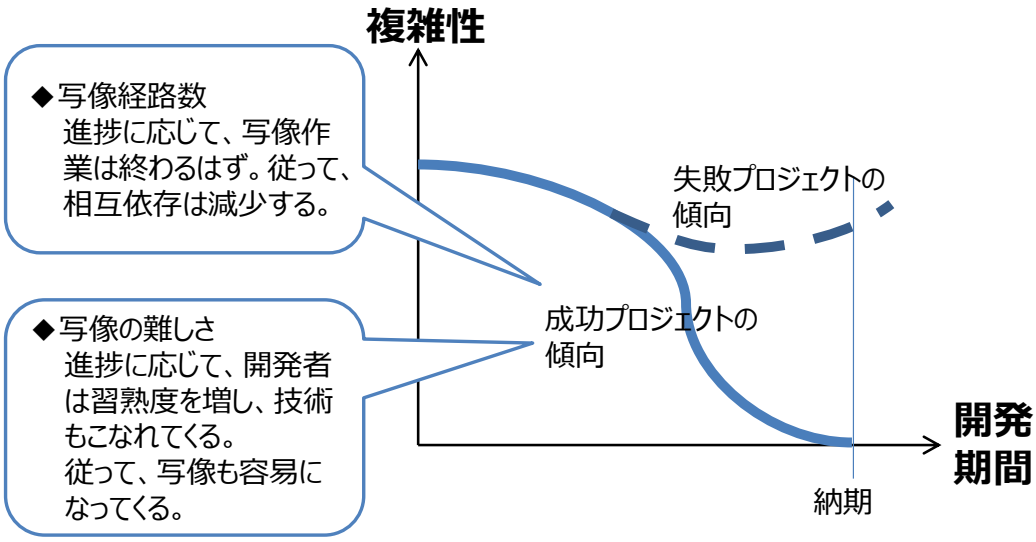


図 4-2 複雑性の時間変化

Figure 4-2 Alteration of Complexity from Moment to Moment

4-3 プロジェクトマネジメントの実践に向けて

複雑性指標によるマネジメントの基本的な考え方は，前ページで述べた通りである．実践する上では，工程が進むにつれてモデルが変化することも十分に考慮しておかなければならない．

工程が進むにつれて，プロジェクトの構成要素がより詳細に明らかになってくる（図 4-3 の $T=t_0$ から t_0+t_1）．はじめは大きい粒度の構成要素でモデルは作成される．工程が進むにつれ，モデルを構成している大きい要素が，更にどのような要素で構成されているかが見えてくる．そして，さらにその詳細化された要素間の関連性（写像経路）も見えてくる．その時，例えば，大きい粒度では 1 本にしか見えなかった写像経路ではあるが，実はふたつの要素間で別々の写像経路で関連性を持っていることを理解するのである．そして更に，詳細化された要素は，プロジェクト内で整理され，サブシステム化（サブグループ化）される（図 4-3 の $T=t_0+t_1$ から $t_0+t_1+t_2$）．プロジェクトはこのような変化を辿る．

この様な過程でモデルの粒度が詳細化されていく．その結果，写像経路数が当然増大する．写像経路数が増加すれば，プロジェクト全体の複雑性も増加する．しかし，この事象は本当にプロジェクトの複雑性が増大していると捉えて良いのだろうか．すなわち，例えば，経験豊富なプロジェクトマネージャーが，ある時点で粒度が細かいモデルを作った．その時の写像経路数は 100 本であった．しかし，若手のプロジェクトマネージ

ャー候補が, 同じプロジェクトをモデルかすると, 写像経路が 10 本のモデルになった. このように同じプロジェクトであるにもかかわらず異なる粒度で表現されることも有り得る（3 章参照）. 時間変化でも同様である. ある時点で 10 本の写像経路数でモデル化した. その後, 粒度を細かく, 100 本のモデルを作成した. この時 10 本と 100 本のモデルの複雑性を比較する術を持たなければならない. 3 章で紹介したように, 複雑性を算出する上でモデルの規格化を行うことで比較評価が可能となる. これにより, 再構築前後でモデルの比較評価が出来ると考えている. その結果, 複雑性の変化をモニタリングすることが可能となり, プロジェクトが順調に進んでいるか否かの判断ができるのである.

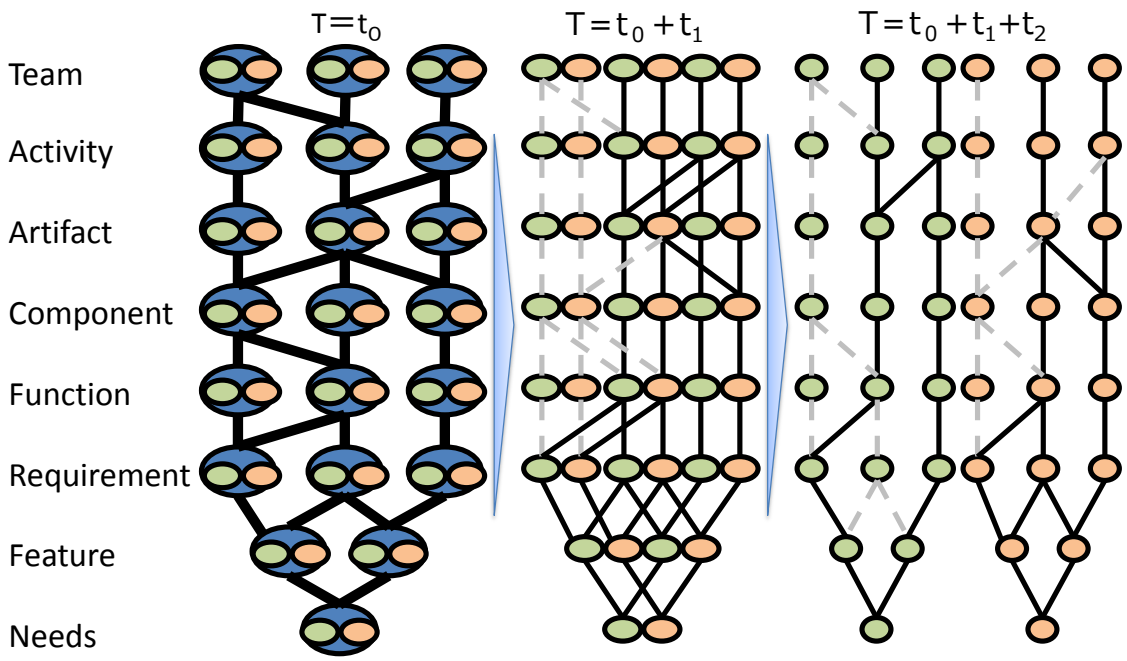


図 4-3 プロジェクトモデルの時間変化

Figure 4-3 Alteration of Project Model from Moment to Moment

次に，通常プロジェクトは開発するシステムの概要が見えてくると，そのシステムをサブシステムに分ける．そのサブシステム化に伴い，プロジェクトもサブシステム毎にサブグループ化され，開発を進める（図 4-3 の $T=t_0+t_1$ から $t_0+t_1+t_2$）．すなわち，プロジェクトをマネジメントする上では，工程が進むにつれてプロジェクト全体を把握しつつ，サブグループの状況も把握していかなければならない．この 2 つの視点を満たすための適切な粒度とはどの程度なのか判断することは難しい．細かすぎるとプロジェクト全体をマネジメントする立場からは，そのマネジメントのコストが増加するだけであり，逆に粒度が大きすぎれば，サブグループ側の十分なマネジメントは出来ない．

このふたつの矛盾を解決するために，2 章で紹介した小行列化を活用する．小行列を使って，鳥の目と虫の目を両立する．すなわち，プロジェクト全体を細分化し，そのサブグループを個々の小行列として管理する．その小行列を見れば，虫の目として，細かく状況を把握することが出来る．また，一方で，小行列化を全てまとめたマトリクスから写像経路数・写像の難しさを算出し，複雑性を求めれば，プロジェクト全体を俯瞰した鳥の目として状況を把握することが出来る．

また一方，理想的には，プロジェクト全体は，例えば 10×10 程度の粒度のプロジェクト行列で状況を把握した方が分かり易い．上記の小行列化したマトリクスでは，小行列が 10×10 のプロジェクト行列では，プロジェクト全体集約すると，サブグループの数にもよるが 100×100 程度のプロジェクト行列になってしまうことも想定される．それでは可読性も悪く，複雑性指標のみに着目する分には問題ないが，どのサブグループで問題が生じているのか分かり難い．従って，図 4-5 のように，プロジェクト全体の目線で見るためのプロジェクトモデル，プロジェクト行列があり，更に個々のサブグループの目線で，個々のサブグループ単位にマネジメントするためのプロジェクトモデルとプロジェクト行列があることが望ましい．そして，プロジェクト全体の目線とサブグループの目線で作成したプロジェクトモデル・プロジェクト行列の関係がトレース可能な状況にあることが望ましいと考えた．言い換えれば，鳥の目として，プロジェクト全体を把握，虫の目としてサブグループの状況を把握，魚の目として，プロジェクト全体・サブグループのトレンドを掴む，これら 3 つの視点を用意することが望ましいということである．

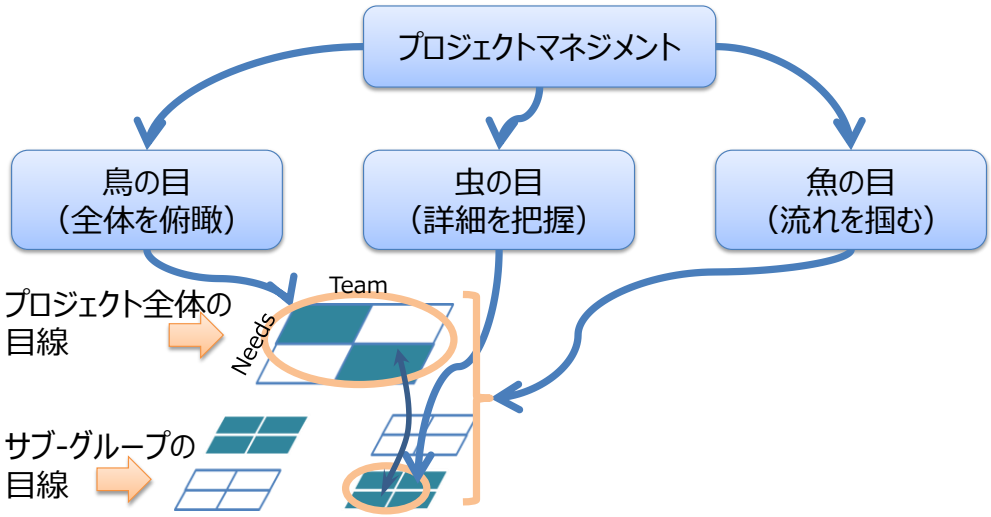


図 4-5 プロジェクトマネジメントの目線

Figure 4-5 View of Project Management

現状，プロジェクト全体の視点，サブグループの視点，それぞれのレイアーに閉じたマトリクスの作り方については，2 章でも述べている．また，各レイアーに閉じたモデル粒度の変化については，3 章でのモデルの規格化方法を用いれば良い．しかし，各レイアーを繋ぐ方法について，まだ説明が出来ていない．各レイアーを繋ぐとは，以下のような意味である．例えば，あるサブグループが 2×2 の粒度でプロジェクト行列が構成されるとき，その中のある Team と Needs 間に 10 本の写像経路があるとしよう．その時，プロジェクト全体を表すプロジェクト行列では，写像経路を何本とするのか，また同様に，写像の難しさを幾つに設定することが正解かという問題である．

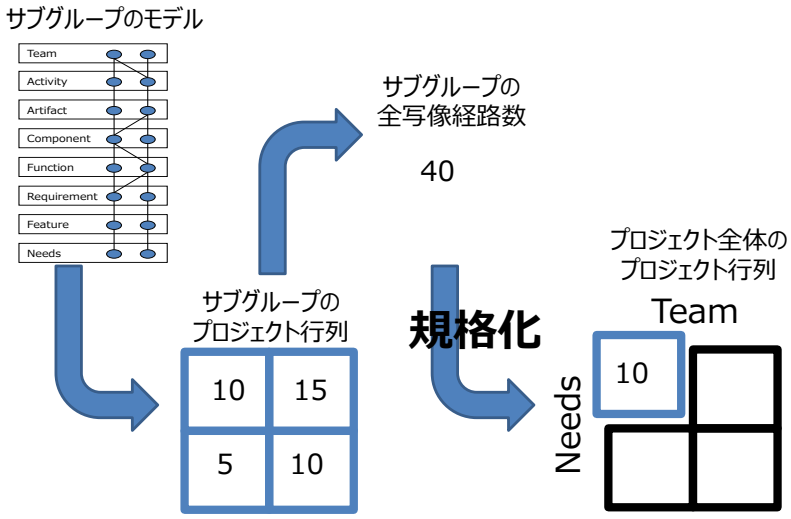


図 4-6 サブグループとプロジェクトの関係

Figure 4-6 The Linkage between Project and Subgroupe

理論的には，以下の手順で算出すればよい．まず，あるひとつのサブグループに閉じて考える．そのサブグループのプロジェクトモデルを考え，写像経路数・写像の難しさを算出し，プロジェクト行列を求める．これにより，サブグループ全体の写像経路数・写像の難しさを求めることが出来る．この手順は，対象がプロジェクト全体としているか，サブグループとしているかの違いだけで，2 章で述べていることと何も変わらない．次に，算出したサブグループの写像経路数と写像の難しさを規格化する．それは 3 章で述べたように，作成したプロジェクト行列の次数（すなわち，モデルの粒度）から，比例計算により，規格化した写像経路数・写像の難しさを求めれば良い．その算出結果を全プロジェクトのプロジェクト行列に該当する箇所にひとつの成分値として組込めば良い．図 4-6 では，サブグループのモデルが，2×2 のプロジェクト行列になるモデルとしている．その写像経路数を求め，規格化し，プロジェクトの行列に埋め込んでいる．同様に各サブグループの写像経路を求め，全ての成分を埋めれば，プロジェクト全体を表すプロジェクト行列が出来上がる．写像経路の難しさについても，同じ手順で算出し，プロジェクト行列を作ればよい．その結果，サブプロジェクトとプロジェクトを繋ぐことが出来るのである．このように，サブグループからプロジェクトの写像経路数・写像の難しさを求めることが出来れば，プロジェクト全体の複雑性指標の時間変化もトレースすることが可能である．その結果，鳥の目・虫の目・魚の目を使ったプロジェクトマネジメントが実現できる．

4-4 複雑性指標を用いたプロジェクトマネジメントの基本的な考え方

以上を踏まえて，筆者が考えるプロジェクトマネジメント方法を説明する．実際のプロジェクト運営の中で，どのように複雑性指標を中心にしたマネジメントを行うか，その基本的な考え方を以下で説明する．説明にあたっては PDCA サイクル（計画(Plan)→実行(Do)→評価(Check)→改善(Action)のサイクル）をシナリオに用いる（図 4-7）．

はじめに，情報写像モデルを作成し，プロジェクトを構成する要素とその関連性を整理する．そして，出来上がった情報写像モデルを元にプロジェクトの行列を作成する．その際には個々の要素の持つ写像の難しさも決定し，プロジェクト全体の行列を作成する．そのプロジェクト行列から，計画当初の複雑性を算出する．その複雑性指標を踏まえ，工期・工数・生産性等の妥当性を検討する．必要に応じて計画段階で再度情報写像モデルの見直しを行い，複雑性指標の低減に取り組むことで工期・工数等を適正な範囲に抑えた計画を立案する．その結果得られた情報写像モデルを元に実際にプロジェクトを始動する．プロジェクトの状況を週単位等，定期的に把握する際には，複雑性指標を活用する．その変化を見ることでプロジェクトの状況を俯瞰する．複雑性の変化が増加傾向にあるならば，それはプロジェクトの状況になんらかの問題が生じている可能性がある．そのため，原因となっている要素や写像経路を洗い出す．そして，情報写像モデルを元に，どのように要素の持つ写像の難しさや写像経路を変化させれば，プロジェクト全体の複雑性を低減することに一番効果的なのかを机上で検討し，プロジェクトの状況を改善する．以降では，PDCA サイクルに沿って，上記内容の詳細を説明する．

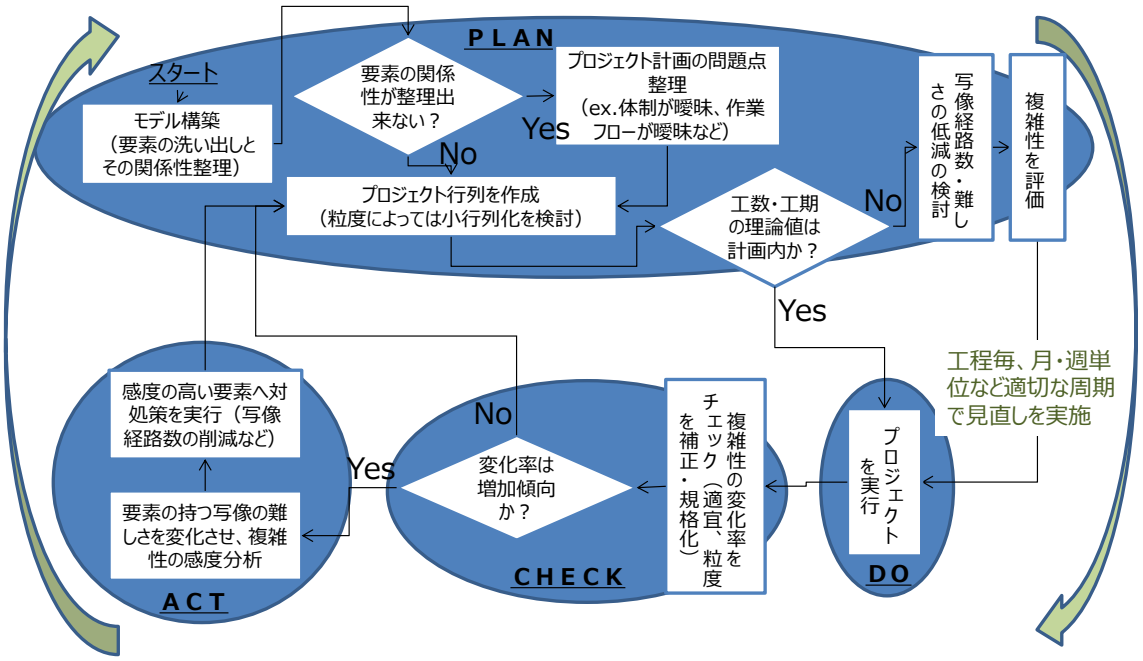


図 4-7 複雑性指標によるプロジェクトマネジメントの PDCA サイクル

Figure 4-7 PDCA Cycle of Project Management by the Complexity Index

◆PLAN

要素の洗い出しとその関係性の整理

プロジェクトの現状分析を行い，構成する要素やその要素間の関係を整理し，情報写像モデルを作成する．場合によっては，小行列化を考慮しながら整理することが必要である．しかし，要素の粒度がまちまち，または関係性が分からない等の理由からモデルを作成できないことも想定される．例えば，作業項目が十分に洗い出されていないため Activity が分からない，要求条件が曖昧なため Requirement が分からない等，そもそもプロジェクト計画に荒さや漏れがあるため，そもそもプロジェクトが開発に着手するレベルに至っていないことが考えられる．情報写像モデルの作成過程の中で，このように課題を浮き彫りにし，その解決を行うことが必要である．それらの課題を整理し，行列を用いてプロジェクトの複雑性指標を算出する．その算出結果に基づいて，工数と工期が計画内に収まるか否か，3 章で述べた理論式により改めて検証する．

情報写像モデルを用いたプロジェクトの改善

情報写像モデルとして示した基本モデルを目指した改善を行う．つまり，プロジェクトを示す行列で考えると，個々の行列を対角化することを目指す際に留意すべき点につ

いて述べる．

プロジェクトの複雑性は，前述の通り，その算出過程の中で複数の行列の積を求めることとなる．その情報写像モデルを行列化した内，ある行列を対角化しても，他の行列の非対角要素を増やしてしまうことがある．その結果，写像経路数の増大や写像経路の難しさの増大が生じてしまうことが想定される．行列の掛け算の特性から，全行列または三角行列が組み込まれた計算の解は必ず全行列または三角行列となる．対角行列を解として得るには各行列を対角行列にする必要がある．従って，ステップ 1 として，写像経路数を求める際に用いる行列（つまり，成分値が 0 or 1）を構成する各行列の関係性を整理し，対角行列に近づけることによって写像経路数を低減する．ステップ 2 として，写像経路の難しさを算出するための各行列成分の中でその成分値の大きい関係に絞り，その値を低減することによって，写像の難しさを低減する．以上によりプロジェクトの複雑性の低減を実現する．

しかし，現実のプロジェクトでは，情報写像モデルで示したような基本プロジェクトの形にすること，つまり，情報写像モデルを行列表現した場合に構成される全ての行列を対角化することは困難である．したがって，例えば各行列を三角行列化することを目的にプロジェクトの設計を改善する（写像経路数の低減）．そして，各要素間の関係性を例えば開発ツール等の導入によって平易なものとし，行列成分値の値を低減する（写像経路の難しさ低減）．それにより全体としての複雑性を低減する案もある．また別案として，プロジェクトを構成する行列の中で，全行列とするのは，1 つまたは 2 つの行列に絞り込み，他の行列を対角化し，且つ，その行列成分（写像経路の難しさ）の値を極力低いものとするような改善も有効である．これは，全行列である箇所の複雑性は高いかもしれないが，対角化した箇所の複雑性を低減することにより，全体の複雑性を下げるという考え方である．ただし，どちらの場合も必ずしも複雑性の低減に有効でないケースも有り得るため，プロジェクトの設計段階でのシミュレーションによる比較検証は必要である．シミュレーションを行い，改善方法の良否を判断し，実行に移る．

◆DO

計画したとおりの形でプロジェクトを実行し，開発を進める．

◆CHECK

本節ではプロジェクトの時間的変化に対して議論を行う．複雑性は開発が進むにつれて低減することが理想である．なぜならば，完全に情報写像が終わった要素間ではその相互依存性が無くなっていくからである．従って，最終的には写像経路数は 0 に収束する．同時に，習熟効果等により写像経路の持つ難しさも時間の変化に伴って低減する．ただし，その難易度は必ずしも最終的に要素値が 0 に収束するとは限らない．例えば，限られた開発期間の中でパッケージソフトの全仕様を把握し，どのような場合でも対処

可能なスキルを習得することは通常難しい.

従って, プロジェクト全体の状況をマネジメントする方法として 2 つの観点が考えられる．1 つ目はプロジェクト全体を俯瞰する視点から，本研究で提案した複雑性を管理指標としてプロジェクトの状況管理を行う. 複雑性の変化率が減少傾向にあるならば順調に開発が進行していることを示し, 変化率が増加傾向にあるならばプロジェクトで何らかの問題が生じている可能性があるという理解である．また 2 つ目として，プロジェクトを構成する個々の要素をマネジメントする観点から, 写像経路数を指標としてプロジェクトの状況管理を行う．プロジェクトの個々の要素間の関係性，つまり写像経路数とその変化率がどのように変化しているか，その状況を捉えることで，個々の要素間での写像が順調か否かを把握することが出来る. 要素間での写像過程が上手く進んでいなければ, 写像元から写像先への写像経路数が減少傾向を示すことは無い. その傾向から, 個々の要素に何かしらの問題が生じていることの兆候を見て取ることが出来る.

また, プロジェクトが進むにつれ, プロジェクトを構成する細かな要素が見えてくる. そのため, プロジェクトモデルをより粒度の細かなものに再構築してみることも適宜必要になってくる．複雑性の精度を向上させる意味でも，粒度の細かなモデルは必要であり，その結果，コストオーバーや納期遅延の抑止に役立つのである.

◆ACT

本節では，複雑性の変化の傾向から，どのようなアクションを取るべきなのか説明する．複雑性の変化率が増加傾向にあるならば，何が問題なのか情報写像モデルから俯瞰し，その問題点を洗い出す．そして直接的にその問題を解決し，複雑性を低減できるならば，その対処を実施する．例えば，Requirement の要素の中に，その内容が曖昧なものがあり，その結果，関連する function への写像経路の難しさが増大している場合ならば，その Requirement を明確にすることに注力すれば良い．しかし，例えば，ある製品を導入した開発中に，ある Requirement に対して，導入した製品(Component)の持つ function では，その Requirement の実現が困難なことが判明した場合を想定する．その場合，写像経路をそのままに保ち，その Requirement を製品の function に合致したものに仕様を変更する案が，対処策として考えられる．また別の対処案としては写像経路を変更し，その製品(Component)が持つ function ではなく，新たな function をプロジェクトの構成要素として起こす案も，対処策として考えられる．後者の対処案のような場合ならば，対処案を想定した情報写像モデルを新たに作成し，その複雑性をシミュレーションすることで効果を予測することも可能である. このように情報写像モデルを中心に, 対処案を検討しその効果を予測することで最善案を選択することが重要である.

そして最後に，マネジメントするために PDCA（Plan-Do-Check-Action）を回す周期についてである．モデルの構成要素の粒度が大きいほど，その周期は長くすべきである．また粒度が小さければ周期は短くすべきである（図 4-8）．粒度が小さいほど，その変化は鋭敏であり，サイクルの周期は短くする必要がある．しかし粒度が大きければ，PDCA サイクルの周期が短くても，状況の変化は然程みられることもなく，そのサイクルを回す作業だけが重いものとなってしまう．すなわち，マネジメントコストだけを無用に膨らましてしまう．そのため，プロジェクトの状況を踏まえて，粒度とサイクルのバランスを考慮し，調整していくことは，本研究で提案した複雑性指標を有効に活用していくために，開発プロジェクトの現場では重要なことである．

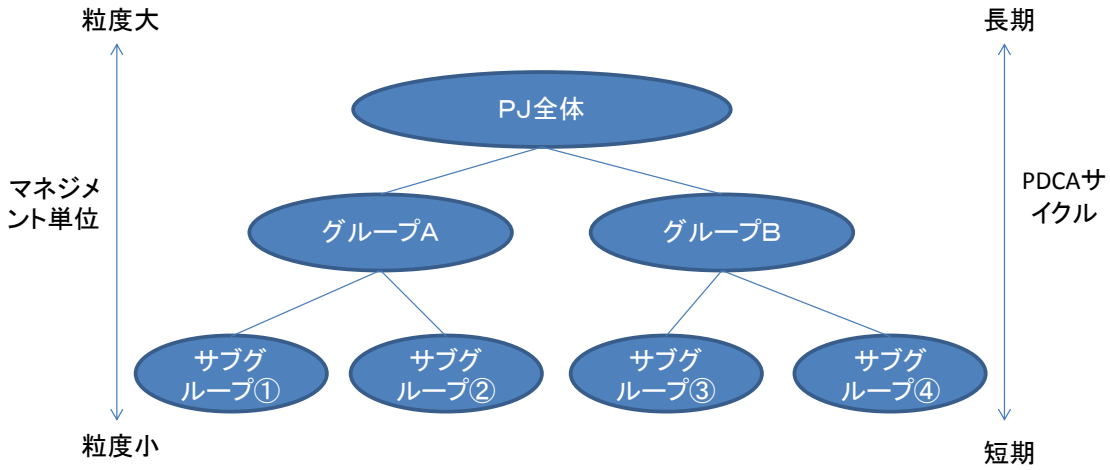


図 4-8 マネジメントの対象粒度と PDCA サイクル

Figure 4-8 Size for Management and PDCA Cycle

4-5実プロジェクトにおけるその他留意事項

実際にプロジェクトへ本マネジメント手法を導入した際に，これまで述べてきたようなマネジメント技術的な課題ではないが，その他（契約条件等）留意しておくべき事項について説明する．

まず，プロジェクトのモデル作成過程において留意すべき点があると考えている．プロセス領域の関係性は，多くの企業では開発標準等が整備されていると考えるので，それを元に関係性を調査し，モデル化することが可能と考えている．実体領域についても，過去のアーキテクチャー等を参照することにより，おおよそのモデルは作成可能であると考えている．しかし，機能領域・顧客領域においては，顧客の思考が強く反映する領域であるため，要素間の整理が進められないケースもあると想定している．何故ならば，顧客には要求工学を知っている訳でもない．従って，自らが Needs から Requirement までの関係をシンプルに整理することは難しい．恐らく，開発側でそれを整理するとしても，要素間の関係はメッシュ構造になってしまうのではないかと考えている．そのため，モデルを作ることは出来ても，写像経路数や写像の難しさの低減に向けて顧客を巻き込むことには大きなパワーを擁してしまうかもしれない．

次に，写像経路の平均作業期間の見積精度をどのように向上すべきかという点である．写像経路の平均作業期間は，精度高く工数・工期を見積もる上で大きな影響を与える変数である．しかし，残念ながら，現状ではその見積り指針等を持っている訳ではなく，あくまでも有識者のスキルに依存し，また幾つかのプロジェクトを経験する中で，そのノウハウを蓄積していくことがひとつの解であると考えている．また，高い精度で見積もるためには細かな粒度でモデルを作成する必要も出てくるであろう．そうなれば，そのモデルを構築することだけで，マネジメントコストが大きく跳ね上がってしまう．作業期間を精度高くイメージできるレベルまでモデルの粒度を細かくしてみることもひとつの解であると考えているはいるものの，有識者のスキルとマネジメントコストのバランスによって，見積精度は決まってしまうのかもしれない．但し，そのような場合でも，PDCA として説明した通り，適宜モデルを見直ししていくことで，開発の初期工程では精度が悪くとも，確実に精度を向上させていくことは出来る．プロジェクトは生き物であり，随時その変化を適切に捉えていくことは重要である．

また最後に，契約上の問題について述べる．開発にあたって請負契約を結び，協力会社へ開発作業を発注することも少なくない．その場合，発注先の企業が写像経路やその難しさを規定し，モデルを作成することに協力を得るにはある程度の交渉が必要であろう．発注先企業は，マネジメントコストの増大を懸念して，難色を示すことも想定しておかなければならない．旧来のやり方に比べて，マネジメントコストが増大するとは考えていないが，慣れ親しんだやり方を変える際には色々な壁が現れてくるものである．その壁を下げるには実績を作り，その良さを理解し浸透させていくしかないと考えている．どちらにしても，発注先の協力を得られるように努力しない限り，写像経路等を把

握することは困難になってしまうことは考慮しておくべきである．

以上，実プロジェクトへこのマネジメント方法を導入する上で想定される課題を説明した．どれもまだ良い解決策を見いだせていない状況である．しかし，ここに挙げた課題は，机上で検討するだけで対処できる種類のものではないと考えている．上記でも説明したように，顧客や発注先の企業側の意志次第で，その対処方法も変わるからである．また見積精度の問題も，多くの経験を積んだキーマンがプロジェクトに居るか否かで対策も変わってしまうものである．従って，課題はあるが積極的に導入を試みる中で解決策を見出していきたいと考えている．

4-6 まとめ

本章で述べたプロジェクトマネジメント方法は，複雑性指標を用いて，実際にマネジメントしていくための基本的な考え方を述べた．基本的な考え方は，本章のはじめにも説明した通りであるが，実証検証を完全に終えた完成したマネジメント方法論ではない．今後は，その完成度を上げ，方法論を完成させる必要がある．そして更に検証データを蓄積し，基本的な考え方に誤りの無いこと，また検討の漏れを確認していきたい．データを蓄積する際にも，事前に全ての課題をクリアすることは不可能であると考えている．一足飛びにパーフェクトを目指すことは難しいと考えるが，ここで述べた考え方を少しずつ取入れて実践し，検証していければと考えている．その結果によって，ソフトウェア開発に携わる方々全体の効率化に貢献できればと考えている．ソフトウェア開発業界全体レベルアップを願って，次章にて本論文を締めくくる．

【第５章】

結論

５．　結論

本研究は，プロジェクトの複雑さを理解し，プロジェクトマネジメントを容易化することを目的とした．具体的には，ソフトウェア開発プロジェクトの成功率の低さを課題とし，その改善を目指して，設計情報の写像経路に着目した複雑性指標を提案した．

まず初めに 1 章にて，プロジェクト開発の現状およびその問題点を先行研究レビューにより，その課題をまとめた．そして，先行研究でも指摘された複雑なプロジェクトのマネジメントに関する限界点について述べた．

次に 2 章では，ソフトウェア開発プロジェクトの要素間の複雑な関係をシンプルに表すモデルを提示し，その有用性を示した．本モデルでは，設計情報の写像経路に着目したプロジェクトの複雑性指標を用いた．そして，その複雑性指標と生産性（1 人あたりの平均作業日数）及び開発工数が相関することを実証的に検証した．これにより，プロジェクトの生産性向上や工数削減を実現するためには，設計情報を写像する経路の冗長性を低減することが必要であり，そのためには提示したモデルが有用であることを示した．そのモデルとは，公理的設計論に基づいて構築され，ソフトウェア開発プロジェクトにおけるプロセスやプロダクトを構成する要素間の関連の複雑さを表している．その複雑性は，設計情報の写像経路数や各経路の写像の難しさから，プロジェクト全体の写像の難しさを数値化したものである．そして提示したモデルの有効性を示すために，実際に 21 のソフトウェア開発プロジェクトからデータを収集することで検証を行った．その結果，複雑性指標と生産性や開発工数との間に強い相関があることが分かった．これらの検証結果から，本研究で示したモデルの複雑性指標を低減することはプロジェクトの生産性向上や工数削減に有効であることを確認した．

3 章では，工数と工期の経験モデルが何故累乗関数で表されるのか，そして，その乗数，指数が何故様々な値を取るのか，理論モデルを用いて明らかにし，理論的に工数と工期の関係式を導出した．その理論モデルはプロジェクト内でやり取りされる設計情報の写像経路と，その写像の難しさを変数としたモデルを用いた．本論文の結果，乗数は写像経路数とその写像の難しさに依存した関数で規定され，指数は写像経路数のみに依存した関数で規定されることが分かった．そして，これらの関数を元に理論的に算出した近似式と，実際のプロジェクトデータとを比較し，本理論モデルが実際のプロジェクトの工数と工期の関係を再現していることを実証検証した．以上により，理論モデルによって，工数と工期の経験式と理論式の整合性を確認し，そのメカニズムを明らかにした．

このように，工数と工期が複雑性（写像経路数と写像の難しさ）によって決定されることを明らかにした．そこで最後に 4 章では，複雑性指標を中心としたプロジェクトマネジメントの展望を説明した．この考え方に基づいて，プロジェクトの複雑性をマネジメントすることによって，工数と工期をコントロールし易くなると考える．従って，本研究の成果は，プロジェクトの成功率改善に向けたマネジメント強化に有用な方策を示

したと考える．

日々の開発作業の中で，自身のプロジェクトの効率化を考えることは中々難しいことである．もし考えられたとしても，実務を任されているエンジニアは，身近な作業のやり方を少し工夫するレベルであり，それが実際にプロジェクトの効率化に結び付くか判断も難しい“業務改善”を行えるのが精々だと考える．しかし本研究では，筆者なりにソフトウェア開発プロジェクト全体を俯瞰し，且つプロジェクトの本質とは何かを考え，今回提案した複雑性指標こそがプロジェクトの効率化を図る一番のキーファクターであると考えた．

一方で課題も残っている． ひとつはマネジメント理論の体系化である．4 章では，基本的な視点・考え方，そして展望を述べたのみで，プロジェクトマネジメントの方法論として体系化したとは言い難い．そこで，実際に本指標を活用したプロジェクトマネジメントを実施する中で，課題やその解決策を整理し，体系化することを目指したい．

そしてもうひとつの課題は，適用範囲の拡大である．今回提案した複雑性指標はソフトウェア開発プロジェクト以外のプロジェクトにも適用できると考えられる．今後は他分野のプロジェクトでの適用も行い，様々なプロジェクトの効率化に貢献していきたい．ソフトウェア開発プロジェクトに限らず，プロジェクトのマネジメントは難しく，納期遅延，予算オーバーの問題は生じている．従って，他業界のプロジェクトへの適用拡大を検証することは有用であると考えている．そして最後に，研究成果の精度向上である．本研究では，汎用性を考慮したプロジェクトで検証を行った．しかし，ゲーム開発プロジェクトに閉じた検証しか出来ておらず，汎用性を十分に検証したとは言い難い．今後は，様々なプロジェクトで検証を行い，本論文成果の汎用性についても実証していきたいと考える．

以上を持って本論文を締めくくる．筆者自身もエンジニアであり，業界の進歩に貢献できることを望んでやまない．本研究により，プロジェクトの成功率向上や，または本研究から新たな発想を得た研究者によって，更に良いアイディアが生まれてくれれば幸いである．

謝辞

本研究は，著者が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後期博士課程在学中に，同研究科前野隆司教授・研究科委員長のご指導の下，実施したものです．本研究にあたり，終始懇切なご指導，ご鞭撻を賜りました．前野隆司教授に厚くお礼申し上げます．また，論文を作成するにあたりご指導とご助言を賜りました副査である慶應義塾大学理工学部高田眞吾教授，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　西村秀和教授，白坂成功准教授に心から感謝申し上げます．そして，本論文の基礎となる査読付原著論文の作成にあたり，貴重なご意見，ご指導を頂きました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 狼嘉彰前研究科委員長，神武直彦准教授，保井俊之特別招聘教授，東京大学大学院情報理工学系研究科システム情報学専攻 牧野泰才講師に深く感謝致します．

また，本研究での実証検証を行うにあたってはイレギュラーズアンドパートナーズ株式会社代表取締役 山本一郎様，チーフディレクター 渡邊謙太様に多大なご協力を頂きました．心より感謝申し上げます．

最後に，著者が勤務しております NTT コムウェア株式会社出身で元 NDS インフォス株式会社常務取締役 島田晃様，株式会社アクセス常務取締役 守矢永司様には，情報システム開発プロジェクトの現状とその問題点を理解する上で，非常に有益な視座を与えて頂きました．両氏に深く感謝致します．

2016 年 3 月

榮谷　昭宏

【参考文献】

参考文献（1 章；緒言に関する参考文献）

1) Standish Group, 2015 CHAOS Report. Retrieved Dec 3, 2015, from https://www.standishgroup.com/

2) Frank, M., Sadeh, A., & Ashkenazi, S.: The relationship among systems engineers' capacity for engineering systems thinking, project types, and project success. Project Management Journal, 42 (5), 31–41 (2011) .

3) White, D., Fortune, J.: Current Practice in project management - an empirical study. International journal of project management 20 (1), 1-11 (2002).

4) Thomas, G., Fernandez, W.: Success in IT projects: a matter of definition? . International journal of project management 26 (7), 733-742 (2008).

5) Boehm, B. W.: Some Future Trends and Implications for Systems and Software Engineering Processes, System Engineering 9 (1), 1-19 (2006).

6) Nathan J. Slegers, Ronald T. Kadish, Gary E. Payton, John Thomas, Michael D. Griffin, and Dan Dumbacher.: Learning from Failure in Systems Engineering: A Panel Discussion, System Engineering 15 (1), 74-82 (2012).

7) Boehm, B. W.: Making a Difference in the Software Century, Computer (IEEE Computer Society) March, 32-38 (2008).

8) Eppinger, S. D.: Innovation at the speed of information. Harvard Business Review 79 (1), 149-158 (2001).

9) Danilovic, M., Sandkull, B.: The use of dependence structure matrix and domain mapping matrix in management uncertainty in multiple project situations. International Journal of Project Management, 23 (3), 193–203 (2005).

10) Trietsch, D., & Baker, R. K.: PERT21: Fitting PERT/CPM for use in the 21st century. International journal of project management 30 (4), 490-502 (2102).

11) Vankoucke, M.: Measure the efficiency of project control using fictitious and empirical project data. International journal of project management 30 (2), 252-263 (2012).

12) Suh, N. P.: Axiomatic Design: Advances and Applications. Oxford University Press, New York (2001).

13) Simon, H. A.: The Sciences of the Artificial. The MIT Press, Cambridge (1996).

14) Danilovic, M. and Browning, T. R.: Managing complex product development projects with design structure matrices and domain mapping matrices, International Journal of Project Management, 25, 300–314 (2007).

15) Steward, D. V.: The design structure system: A method for managing the design of complex systems. IEEE Transactions on Engineering Management, 28 (3), 71–74 (1981).

16) Eppinger, S. D.: Model-based approaches to managing concurrent engineering. Journal of Engineering Design, 2 (4), 283–290 (1991).
17) Eppinger, S. D.: Using the design structure matrix to estimate product development time. Proc. ASME Design Engineering Technical Conferences (1998).
18) Shi, Q., & Blomquist, T.: A new approach for project scheduling using fuzzy dependency structure matrix. International journal of project management 30 (4), 503-510 (2012).
19) Eppinger, S. D., Salminen, V.: Patterns of product development interactions. International Conference on Engineering Design ICED 01, August 21–23 (2001).
20) Sosa, M. E., Eppinger, S. D. and Rowles, C. M.: Are your engineers talking to one another when they should? Harvard Business Review, 85 (11), 133–142 (2007).
21) Lindeman, U., Maurer, M. and Braun, T.: Structural Complexity Management– An Approach for the Field of Product Design. Springer, Berlin (2009).
22) Kaplan, R. S. & Norton, D. P.: Balanced Scorecard: Translating Strategy into Action. Boston, Mass., Harvard Business School Press, 322 (1996).
23) Vidal, A. L., Marle, F., Bocquet, C. J.: Measuring project complexity using the Analytic Hierarchy Process. International journal of project management, 29 (6), 718-727 (2011).
24) Bosch-Reveldt, M., Jongkind, Y., Mooi, H., Bakker, H., Verbraeck, A.: Grasping project complexity in large engineering projects: The TOE (Technical, Organizational and Environmental) framework. International journal of project management, 29 (6), 728-739 (2011).
25) Shannon, C. E.: The mathematical theory of communication. Bell Systems Technological Journal, 27, 379-423 (1951).

参考文献（2 章；複雑性指標導出に関する参考文献）

1） Brooks, F. P.: The Mythical Man-Month, Anniversary edition. Pearson Education, MA （2010).

2） 犬塚篤： ソフトウェア開発における情報処理の多面性. 電子情報通信学会. SWIM, ソフトウェアインタプライズモデリング, 105 (422), 9-14 (2005).

3） 犬塚篤: 顧客ニーズの共有コストに関する考察. 日本経営学会誌, 14, 43-54 (2005).

4） Sosa, M. E., Eppinger, S. D. and Rowles, C. M.: Are your engineers talking to one another when they should? Harvard Business Review, 85 (11), 133–142 (2007).

5） von Hippel, E.: “Sticky information” and the locus of problem solving: Implications for innovation. Management Science, 40 (4), 429–439 (1994).

6） Draft, R. L., Lengel, R. H.: Organizational information requirements, media richness, and structural design. Management Science, 32 (5), 554–571 (1986)).

7） Eppinger, S. D.: Model-based approaches to managing concurrent engineering. Journal of Engineering Design, 2 (4), 283–290 (1991).

8） Eppinger, S. D.: Using the design structure matrix to estimate product development time. Proc. ASME Design Engineering Technical Conferences (1998).

9） Eppinger, S. D.: Innovation at the speed of information. Harvard Business Review, 79 (1), 149–158 (2001).

10) Eppinger, S. D., Salminen, V.: Patterns of product development interactions. International Conference on Engineering Design ICED 01, August 21–23 (2001).

11) Danilovic, M., Sandkull, B.: The use of dependence structure matrix and domain mapping matrix in management uncertainty in multiple project situations. International Journal of Project Management, 23 (3), 193–203 (2005).

12) Danilovic, M., Browning, T. R.: Managing complex product development projects with design structure matrices and domain mapping matrices. International Journal of Project Management, 25, 300–314 (2007).

13) Steward, D. V.: The design structure system: A method for managing the design of complex systems. IEEE Transactions on Engineering Management, 28 (3), 71–74 (1981).

14) 丸山勝久： 日本のソフトウェア工学の今と未来. 情報処理, 49 (7), 793-798 (2008).

15) White, D., Fortune, J.: Current practice in project management–An empirical

study. International Journal of Project Management 20 (1), 1-11 (2002).
16) Sakaedani, A., Yasui, T.: Complexity Management by Using Project Matrix. Proc. GLOGIFT2010 Yokohama, Japan, Paper#1569319363 (2010).
17) Sakaedani, A., Yasui, T.: Inter-element relations in the configured systems: Second dimension of the system complexity from the case study of the Japan's anime industry. Proc. APCOSE 2010 Keelung, Taiwan, Paper #272 (2010).
18) 榮谷昭宏，狼嘉彰，神武直彦： 高品質なプロジェクトマネジメントを実現するトレーサビリティ・マトリックスの構築． シンセシオロジー，5 (1), 1-16 (2012).
19) Sakaedani, A., Yasui, T., Shirasaka, S. and Maeno, T.: A New approach to component reuse in multi-project management by using an information-centric project model. Proc. The 14th International DSM Conference, 301–313 (2012).
20) Sakaedani, A., Yasui, T. and Shirasaka, S.: Empirical validation of information centric project management: Enhancement of problem-solving capability by reduction of project complexity. Proc. The 6th International Conference on Project Management (ProMAC2012), 751–758 (2012).
21) Lindeman, U., Maurer, M. and Braun, T.: Structural Complexity Management– An Approach for the Field of Product Design. Springer, Berlin (2009).
22) Malik, F.: Strategie des Managements komplexer System. Bern, Haupt (2003).
23) Ehrlenspiel, K.: Integrierte Produktentwicklung - Methoden fur Prozessorganisation, Produktersterllung und Konstruktion. 3rd Ed. Munchen, Hanser (2007).
24) Suh, N. P.: Axiomatic Design: Advances and Applications. Oxford University Press, New York (2001).
25) Leffingwell, D., Widrig, D.: Managing Software Requirements A Unified Approach. Addison-Wesley, MA (2000).
26) Forsberg, K., Mooz, H. and Cotterman, H.: Visualizing Project Management Models and Frameworks for Mastering Complex Systems. 3rd Ed. John Wiley&Sons INC (2005).

参考文献（3 章；工数－工期の関係に関する参考文献）

1) Boehm, B. W.: Software Engineering Economics, Prentice-Hall, Engle-wood, Cliffs, N.J. (1981).

2) Boehm, B. W. et al.: Software Cost Estimation with COCOMO II, Prentice Hall (2000).

3) Walston, C. E. and Felix, C. P.: A method of programming measurement and estimation, IBM Systems Journal, 16 (1), 54–73 (Mar. 1977).

4) Reifer, D.: Estimating web development costs: there are differences, Cross talk, June (2002).

5) ソフトウェア開発データ白書 2012-2013，独立行政法人 情報処理推進機構（IPA）技術本部ソフトウェア・エンジニアリング・センター（SEC），入手先 http://sec.ipa.go.jp/users/publish/SEC-TN12-002.pdf （参照 2014/08/11）.

6) ユーザー企業ソフトウェアメトリックス調査【調査報告書】2011 年度版，2011 年 9 月，社団法人日本情報システム・ユーザー協会，入手先 http://www.juas.or.jp/servey/library/pdf/11swm.pdf （参照 2014/08/11）.

7) 経済調査研究所研究レポート 2010 ソフトウェア開発データリポジトリの分析，2010 年 7 月，財団法人経済調査会経済調査研究所，入手先 http://www.zaikeicho.or.jp/pdf/software metrics/pdf/ERA Analysis of Software Projects Data Repository 201007.pdf （参照 2014/08/11）.

8) 戸田航史，松本健一，押野智樹，高橋明彦：ソフトウェア開発における適正工期に関する分析，経済調査研究レビュー，7, 37-46 (2010).

9) Bailey, J. W. and Basili, V. R.: A meta-model for software development resource expenditures, Proc. 5th International Conference on Software Engineering (1981).

10] Kemerer, C. F.: An Empirical Validation of Software Cost Estimation Models, Communications of the Association for Computing Machinery, 30 (5), 416–429 (1987).

11) Putnam, L. H.: A general empirical solution to the macro software sizing and estimating problem, IEEE Trans. Softw. Eng., 4 (4), 345–361 (1987).

12) Steward, D. V.: The design structure system: A method for managing the design of complex systems, IEEE Trans. Engineering Management, 28 (3), 71–74 (1981).

13) Sosa, M. E., Eppinger, S. D. and Rowles, C. M.: Are your engineers talking to one another when they should?, Harvard Business Review, 85(11), 133–142 (2007).

14) Eppinger, S. D.: Model-based approaches to managing concurrent engineering, Journal of Engineering Design, 2 (4), 283–290 (1991).

15) Eppinger, S. D.: Using the design structure matrix to estimate product

development time, Proc. ASME Design Engineering Technical Conferences (1998).
16) Eppinger, S. D.: Innovation at the speed of information, Harvard Business Review, 79 (1), 149–158 (2001).
17) Eppinger, S. D. and Salminen, V.: Patterns of product development interactions, International Conference on Engineering Design, ICED '01, August 21–23 (2001).
18) Danilovic, M. and Sandkull, B.: The use of dependence structure matrix and domain mapping matrix in management uncertainty in multiple project situations, International Journal of Project Management, 23 (3), 193–203 (2005).
19) Danilovic, M. and Browning, T. R.: Managing complex product development projects with design structure matrices and domain mapping matrices, International Journal of Project Management, 25, 300–314 (2007).
20) Lindeman, U., Maurer, M. and Braun, T.: Structural Complexity Management – An Approach for the Field of Product Design, Springer, Berlin (2009).
21) Forrester, J. W.: Industrial Dynamics, MIT Press, MA. (1961).
22) Lyneis, J. M. and Ford, D. N.: System Dynamics Applied to Project Management: A survey, assessment, and directions for future research, System Dynamics Review, 23 (4), 157–189 (2007).
23) Abdel-Hamid, T.: The Dynamics of Software Development Project Management, Doctoral Thesis, MIT, Cambridge, MA. (1984).
24) Cooper, K.: Naval Ship Production: A Claim Settled and a Framework Built, Interfaces, 10 (6), (1980).
25) Ford, D. and Sterman, J.: Dynamic Modeling of Product Development Processes, Working Paper, 3943-97, MIT Sloan School of Management, Cambridge, MA. (1997).
26) Homer, J., Sterman, J., Greenwood, B. and Perkola, M.: Delivery time reduction in pulp and paper mill construction projects, Proc. 1993 International System Dynamics Conference, The System Dynamics Society, Cancun, Mexico (1993).
27) Rodrigues, A. and Bowers, J.: System dynamics in project management: a comparative analysis with traditional methods, System Dynamics Review, 12 (2), 121–139 (1996).
28) Lyneis, J. M., Cooper, K. G. and Els, S. A.: Strategic management of complex projects: a case study using system dynamics, System Dynamics Review, 17 (3), 237–260 (2001).
29) Park, M. and Pena-Mora, F.: Dynamic change management for construction: introducing the change cycle into model-based project management, System Dynamics Review, 19 (3), 213–242 (2003).
30) Taylor, T. and Ford, D. N.: Tipping point failure and robustness in single

development projects, System Dynamics Review, 22 (1), 51–71 (2006).
31) Lyneis, J. M. and Ford, D. N.: System dynamics applied to project management: A survey, assessment, and directions for future research, System Dynamics Review, 23(2/3), 157–189 (2007).
32) Heemstra, F. J.: Software cost estimation, Information & Software Technology, 34 (10), pp.627–639 (1992).
33) Boehm, B. W. and Papaccio, P. N.: Understanding and Controlling Software Costs, IEEE Trans. Softw. Eng., 14 (10), 1462–1477 (1988).
34) 榮谷昭宏，牧野泰才，前野隆司：設計情報の写像経路に着目したソフトウェア開発プロジェクトの複雑性指標の提案，情報処理学会論文誌，55 (5), 1453-1470 (2014).
35) Suh, N. P.: Axiomatic Design: Advances and Applications, Oxford University Press, New York (2001).
36) Project Management Institute, Inc.：プロジェクトマネジメント知識体系ガイド（PMBOK ガイド）第 4 版(2008).

参考文献（4 章；複雑性指標を用いたプロジェクトのマネジメントの展望に関する参考文献）

1) Eppinger, S. D.: Innovation at the speed of information. Harvard Business Review 79 (1), 149-158 (2001).

2) Danilovic, M., Sandkull, B.: The use of dependence structure matrix and domain mapping matrix in management uncertainty in multiple project situations. International Journal of Project Management, 23 (3), 193–203 (2005).

3) Trietsch, D., & Baker, R. K.: PERT21: Fitting PERT/CPM for use in the 21st century. International journal of project management 30 (4), 490-502 (2102).

4) Vankoucke, M.: Measure the efficiency of project control using fictitious and empirical project data. International journal of project management 30 (2), 252-263 (2012).

5) Suh, N. P.: Axiomatic Design: Advances and Applications. Oxford University Press, New York (2001).

6) Simon, H. A.: The Sciences of the Artificial. The MIT Press, Cambridge (1996).

7) Danilovic, M. and Browning, T. R.: Managing complex product development projects with design structure matrices and domain mapping matrices, International Journal of Project Management, 25, 300–314 (2007).

【研究業績】

研究業績

学術雑誌掲載論文

① 榮谷昭宏，狼嘉彰，神武直彦： 高品質なプロジェクトマネジメントを実現するトレーサビリティ・マトリックスの構築． シンセシオロジー，5 (1), 1-16 (2012).

② 榮谷昭宏，牧野泰才，前野隆司： 設計情報の写像経路に着目したソフトウェア開発プロジェクトの複雑性指標の提案． 情報処理学会論文誌，55 (5), 1453-1470 (2014)

③ 榮谷昭宏，高田眞吾，前野隆司： 写像経路に着目したソフトウェア開発プロジェクトモデルを用いた工数－工期の理論モデル構築 情報処理学会論文誌，57 (2), (2016)（掲載準備中）

国際会議論文（査読付き full-length papers）

① Sakaedani, A., Yasui, T.: Complexity Management by Using Project Matrix. Proc. GLOGIFT2010 Yokohama, Japan, Paper#1569319363 (2010) .

② Sakaedani, A., Yasui, T., Shirasaka, S. and Maeno, T.: A New approach to component reuse in multi-project management by using an information-centric project model. Proc. The 14th International DSM Conference, 301–313 (2012).

国際会議論文（Abstract 査読付き papers）

① Sakaedani, A., Yasui, T.: Inter-element relations in the configured systems: Second dimension of the system complexity from the case study of the Japan’s anime industry. Proc. APCOSE 2010 Keelung, Taiwan, Paper #272 (2010) .

② Sakaedani, A., Yasui, T. and Shirasaka, S.: Empirical validation of information centric project management: Enhancement of problem-solving capability by reduction of project complexity. Proc. The 6th International Conference on Project Management (ProMAC2012), 751–758 (2012).