東京大学大学院新領域創成科学研究科

人間環境学専攻

平成 30 年度

修士論文

実績データに基づく遅延と手戻りを考慮したプロジェクト工期見積もり手法の開発

2019 年 2 月 7 日提出

指導教員　稗方　和夫　准教授　　印

学生証番号　47-176732

王　汝佳

目次

図目次

表目次

第1章 序論

1.1 背景

ここ数十年間の経済と科学技術の著しい発展に伴い、大規模プロジェクトは、製造業や建設業、および情報産業など、様々な業界において一般的となった。通常、ある程度の規模のあるプロジェクトでは、対象とする製品や技術自体の構造が複雑であるのみならず、作業手順の選定や人員配置の仕組みも複雑なシステムになっていることが多い。このような複雑なシステムにおいては、些細な変化でも、システム全体に多大な影響をもたらすことがあるため、プロジェクトの実行において、高いリスクが潜んでいる場合が多い。一般的に、プロジェクトの成果を大きく左右するような事象は、不確実性（Uncertainty）と呼ばれている。プロジェクトの不確実性をより正確に把握し、それに対して、適切な管理を行うために、多種多様な視点からの試みがなされてきた。

研究を深める前提として、不確実性をより正確に把握するために、不確実性の中身を分類し、確実に定める試みがなされてきた。例えば、Earl ら[1]の定義では、プロジェクトの不確実性は、既知なる不確実性（Known uncertainties）、未知なる不確実性（Unknown uncertainties）、データによる不確実性（Uncertainties in the data）および記述による不確実性（Uncertainties in the description）の 4 つのカテゴリーに分類されている。既知なる不確実性は、過去に起きたことがあり、かつ中身を確認することができる事象を指し、未知なる不確実性は、完全に予想外の事象（大きな自然災害やテロ事件など）を指している。また、データによる不確実性と記述による不確実性とは、それぞれ、データの完全性、一致性、正確性などのブレからなる不確実性と、記述要素やスコープの選定などにおける曖昧さからなる不確実性のことである。さらに、具体的なプロジェクトの種類に応じた不確実性の定義の例として、T. R. Browning[2]は、プロダクト開発（PD, Product Development） プロジェクトの工期について、意図的な繰り返し作業（Intentional iterations）、非意図的な繰り返し作業（Unintentional iterations）、作業セットの完全性（Activity set completeness）、作業のフレキシビリティ（Activity flexibility）、作業工数の変動（Activity and subprocess length and variance）、不確実性を軽減するために行った行動（Uncertainty reduction actions）、および未知なる事象（Unknown unknowns）の不確実性要因を羅列した。

これらの不確実性に対する管理手法として、ポリシーやプロトコルに基づく社会科学的なアプローチ、シミュレーションを代表とする数理モデルに基づく方法、あるいは両者を組み合わせた方法などが提案された。例えば、M. T. Pich ら[3]は、命令（Instructionism）、学習（Learning）、選択（Selectionism）の3つの基本的な管理ストラテジーを定義し、プロジェクト情

報の充実程度に対応するストラテジーの選び方を提案した。また、数理モデルを中心とした諸々の管理手法の中で、シミュレーションは重要な位置を占めており、実プロジェクトの進行に影響しない前提で、リソースの管理や、工期及びコストの予測などができるため、特に実用性が高いと考えられている。

しかし、構成が複雑なプロジェクトに対して、シミュレーションを行うためには、様々な入力情報を要する場合が多い。特に、入力における不確実性に関する部分では、専門家の経験による判断に依存したパラメータの決め方が一般的である。不確実性のモデリング過程における主観性を減じるために、原因を調査する分析手法や、不確実性アンケートによる集計方法などが提案されているが、これらの手法はまた大量な人力と作業を必要とするため、実プロジェクトでの運用が限られている。

1.2 目的

前節で述べた、不確実性による、プロジェクトに対する意思決定の難しさ、および既存手法の局限性を踏まえて、本研究では、実績データに基づき、手戻りと遅延を考慮したプロジェクト工期の見積もりを行う手法を提案することを目的とする。具体的には、下記の 3 点である。

1. 工期見積もりで使用する、不確実性を含めたシミュレーションモデルを提案する。
2. 実績データから、シミュレーションモデルの不確実性パラメータを抽出する手法を提案する。
3. 抽出したパラメータ、および必要とする事前情報を、提案したシミュレーションモデルに入力し、見積もり工期を出力する。

また、ケーススタディでは、シミュレーションによる工期見積もりの結果に基づき、プロジェクトを実行する前に、リソース情報に対する検討が実際に可能であることを示す。

1.3 本論文の構成

本論文の構成は以下の通りである:

第 1 章では、本研究の背景と目的について述べた。

第 2 章では、関連研究について紹介する上、本研究の位置付けと新規性を述べる。

第 3 章では、提案手法の詳細について説明する。第 3 章の前半では、シミュレーションモデルについて説明し、後半では、不確実性パラメータの抽出手法について説明する。

第 4 章では、2 つのケーススタディについて記述する。ケーススタディ1では、提案手法のプログラム上での正しい挙動を検証し、またシミュレーションモデルの妥当性について記述する。ケーススタディ 2 では、提案手法のデモンストレーションを行う。

最後に、第 5 章と第 6 章では、それぞれ、本研究における考察と結論について述べる。

第2章 関連研究

2.1 はじめに

本章では、初めにプロジェクトに対するモデリング手法とシミュレーションモデル、およびそれらの運用について、代表的な研究をまとめる。次に、本研究の目的に関連する、既存モデルにおける不確実性パラメータの設定手法について述べる。最後に、上記の内容を踏まえた上で、本研究の位置付けと新規性を示す。

2.2 大規模プロジェクトのモデル化

2.2.1 基盤とするモデリング手法

大規模複雑プロジェクトに対するモデル化は、プロジェクトを管理するための基本であり、1950 年代頃から、プロジェクトマネジメントの分野、およびシステムズエンジニアリングの分野において、いくつかの基本的なモデリング手法が提案されている。以下では、代表的な例として、WBS (Work Breakdown Structure)、QFD (Quality Function Deployment)と DSM (Design Structure Matrix)の手法について紹介する。

WBS とは、大きなプロジェクトを、レベルに応じて細分化する手法であり、1960 年代に提案され、アメリカ政府により正式に採用されている[4]。WBS によるプロジェクトの細分化は、行動を中心にした作業内容ではなく、階層的な物事を中心としている。WBS の手法は、開発プロジェクトにおける開発対象の構造、情報システムにおけるデータ構造やユーザ需要、または作業人員の組織的構造など、様々な側面から運用することが可能である。QFD とは、1966 年に日本で体系化された管理手法であり[5]、顧客の品質に対する需要を、構成ユニットに応じて細分化し、開発するための技術や、所要のコストなどを洗い出すことができる。DSM は、D. V. Steward[6]により提案されたプロダクトデザイン手法であり、1990 年代後半から、マサチューセッツ工科大学の S. D. Eppinger 教授を中心に、広く運用された[7]。DSM では、2 次元マトリクスを用いて、プロダクトを細分化し、各パーツの詳細な依存関係を定義できる。この手法の応用として、同じく DSM という略称をもつ Dependency Structure Matrix の手法も、多く使われている。Dependency Structure Matrix では、プロジェクトの各作業の手順、および必要とする情報の伝達ルートを定義することができる。また、マトリクスの次元数を増やすことによって、作業の工数や工期、詳細な依存関係、手戻りの確率など[7]、様々な情報を表現することができる。

上記の手法により、プロジェクトの細分化と作業の洗い出しが可能になる。これらの情報を踏まえて、CPM (Critical Path Method)、PERT (Program Evaluation and Review Technique)、GERT (Graphical Evaluation and Review Technique)などのスケジューリング手法が提案された。CPM は、プロジェクトの各アクティビティ、それらの依存関係と所要時間を用いて、プロジェクトを完了するための最短時間を算出する手法であり、デュポン社により 1950 年代に提案された[8]。また、PERT とは、冷戦期のポラリス潜水艦発射弾道ミサイルプロジェクトの一環として、1958 年に提案されたスケジュール管理手法である[9]。PERT では、計算単位であるタスクの工期見積もり、および CPM と類似する最長経路の計算からなる一連の分析手法であり、CPM と併せて、PERT/CPM 手法として、広く知られている。GERT とは、PERT/CPM の上で、依存関係を表すネットワークロジックとアクティビティの工期について、確率要素を加えた分析手法であり、1966 年に、A. A. B. Pritser により、アメリカ航空宇宙局に向けて開発された[10]。古典的な PERT/CPM 手法と比較して、GERT は、それ程広く運用されてはいないが、不確実性の概念を最初に取り入れたモデルとして、その後のプロジェクトの不確実性に関する研究の先駆けとなった手法である。

2.2.2 シミュレーションモデル

2.2.2.1 離散イベントシミュレーション

プロジェクトの実行を計算機上で模擬し、パフォーマンスや実行結果を予測するために、離散イベントシミュレーションがよく使われる。離散イベントシミュレーションとは、数理モデルの解析手法の一種であり、自然科学から社会科学まで、幅広い領域で運用されている。離散イベントシミュレーションでは、物事の変化を離散的な事象として捉え、ステップ毎にシステムの状態の変化を処理している。またプロジェクトマネジメントの分野では、プロジェクト進行中の不確実性を模擬するために、確率的に特定の事象を起こすシミュレーションモデルを取り上げる場合が多い。

離散イベントシミュレーションを用いたモデルの代表的な例として、VDT (Virtual Design Team)[11]が挙げられる場合が多い。VDT のモデルでは、作業間依存関係と組織構造の相互的影響、およびそれらのプロジェクトに与える影響をテーマとしている。そのほかに、DSM を基盤とするシミュレーションモデルもいくつか提案された。例えば、S. D. Eppinger ら[12]は、マルコフ連鎖の確率過程を想定し、異なる作業間の遷移をシミュレーションするモデルを提案した。また、T. R. Browning ら[13]は、並行作業と手戻りを考慮し、プロジェクトの工期と

全体的なコストを見積もるためのシミュレーションモデルを提案している。

2.2.2.2 資源制約付きスケジューリング問題

プロジェクトにおけるモデルベースの研究の 1 分野として、プロジェクトの人員、スペース、機械などの必要資源を考慮する、資源制約付きスケジューリング問題（RCPSP, The Resource-Constrained Project Scheduling Problem）がある。資源制約付きスケジューリング問題は古典的な最適化問題であり、定められた作業と有限な資源に対して、様々な制約のもとで、最適な解を求める。資源制約付きスケジューリング問題に対して、解析的に最適解を求めることは非常に困難であるため、近似アルゴリズムや、前述した離散イベントシミュレーション、もしくは両者の組み合わせを用いて、解を求める場合が多い。R. Kolisch と S. Hartmann[14]は、資源制約付きスケジューリング問題を解くための近似アルゴリズムについて、分類と分析を行った。また、P. Brucker ら[15]は、具体的なモデルを含めて、資源制約付きスケジューリング問題の最適化手法についてまとめている。

実世界のプロジェクトをモデリングした上で、資源制約付きスケジューリング問題を用いて、作業の優先順ルールや資源の配置ルールに対して、検討を行う研究も多数存在する。こういった資源の配置ルールは、ディスパッチングルールと定義される。M. Montazeri と L. N. Van Wassenhove[16]は、古典的なディスパッチングルールについてまとめている。ディスパッチングルールの例として、締め切りの近い作業を優先とする SPT (Shortest Processing Time) のルールや、余裕時間の短い作業を優先とする Slack time rule がある。その上で、B. Grabot と L. Geneste[17]は、ファジィ集合を用いて、ディスパッチングルールの組み合わせの効果について研究を行っている。

なお、スケジューリング問題の一般的な出力結果として、ガントチャートが多く用いられる。ガントチャートとは、1910 年代に経営コンサルタントの H. L. Gantt[18]により考案された、作業スケジュールを表現するための図表であり、プロジェクトにおける各作業の期間を表現することができ、21 世紀の現在においても、頻繁に用いられる手法の 1 つである。

2.2.2.3 不確実性を考慮したシミュレーションモデル

第 1 章で記述した通り、不確実性はプロジェクトの一部であるため、モデルの確率的要素として考慮することもある。特に、代表的な不確実性として、繰り返し作業、あるいは手戻り作業が強調される場合が多い。ここで、D. C. Wynn と C. M. Eckert [19]は、繰り返し作業全般に関する研究を、研究スコープと手法に分けてまとめ、繰り返し作業に対する管理の必要性を強調した。

手戻りを含んだ不確実性要素を考慮したシミュレーションモデルの例として、2.2.2.1 でも紹介した S. D. Eppinger ら[12]と T. R. Browning ら[13]により提案されたシミュレーションモデルがある。S. D. Eppinger ら[12]のモデルでは、並行作業がなく、作業が1つずつ順に実行され、1つの作業が完了する時点で、他のタスクへの遷移が可能となる。上流タスクに対する遷移は、手戻りを意味する。T. R. Browning ら[13]のモデルは、T. R. Browning[2]の 6 章に提案されたモデルをまとめたものであり、作業の並行作業を考慮した上で、作業期間の分布と作業間の手戻りと手戻りによる影響をモデリングしている。さらに、S. Cho[20]は、繰り返し回数によって、変化する手戻り確率を導入し、リソースに対する考慮も含め、T. R. Browning ら[13]のシミュレーションモデルを改善した。

2.3 シミュレーションモデルの運用

本節では、シミュレーションモデルの実際の運用について、関連する研究をまとめる。M. Jahangirian ら[21]は、生産とビジネスにおけるシミュレーションの運用についてまとめているが、本節では、具体的に、プロジェクト早期の意思決定、プロジェクト構造、組織構造および作業現場の管理の 4 つの観点からピックアップし、代表的な関連研究を紹介する。

2.3.1 プロジェクト早期の意思決定に対する支援

シミュレーションモデルの重要な用途の 1 つとして、プロジェクト初期における意思決定の支援がある。ここでいうプロジェクト初期とは、プロジェクトを実際に進行する以前の、プロジェクトの工期見積もり、実行における基本的な方針、人員配置の大枠、などを決定する段階を意味する。

プロジェクトの各作業の構成と人員配置に関する全般的な意思決定は、一般的に、プロジェクトデザインと定義する場合が多い。プロジェクトデザインで、シミュレーションモデルを運用する例として、B. R. Moser[22]は、国際共同プロジェクトに対して、シミュレーションツール「Teamport」を開発した。Teamport では、時間帯や文化が異なる国際チーム間における複雑な依存関係、およびプロジェクト進行中にあり得る様々な不確実性が考慮され、総合的なエージェントベース（各要素や主体の行動を中心とする手法）のシミュレーションモデルを搭載し、プロジェクトのアクティビティ構成から組織構造まで、入力要素を調整し、シミュレーションすることが可能である。また、S. Salimi ら[23]は、高性能計算（HPC, High Performance

Computing）を用いて、橋梁建設における総合的なシミュレーションモデルを提案し、最適解を探索するための遺伝的アルゴリズムを取り入れている。

その他の基本的な意思決定に対して、シミュレーションを用いた研究も多く存在する。例えば、情報システムの開発プロジェクトでは、開発管理手法が大きくプロジェクトの進行と結果を左右するため、緻密な考慮が必要とされている。T. Mitsuyuki ら[24]は、仕様変更とバグフィックスによる手戻りを考慮し、ウォーターフォール開発、アジャイル開発、および 2 つの混合開発手法から、適切な戦略を選定できるシミュレーションモデルを開発した。また、製造業の例として、満行ら[25]は、船舶建造工程における新規設備の導入における意思決定を、支援するシミュレーションモデルを開発した。その他に、特殊な条件や状況に応じて、意思決定を支援するシミュレーションモデルの例として、小林ら[26]は、生産性とエネルギー消費量の制約を考慮するマネジメント手法を提案し、その手法の検証を行っている。満行ら[27]は、電力ピークカットがある前提で、船舶建造作業の計画立案の意思決定支援に対して、遺伝的アルゴリズムを組み合わせ、シミュレーションモデルを提案した。

2.3.2 プロジェクト構造に対する検討

具体的に、プロジェクト構造（プロジェクトの細分化や各作業の優先順位）に対して、検討を行うシミュレーションモデルも多く提案されている。例えば、J. F. Maier ら[28]は、一般的なプロジェクトの設計プロセスに対して、意図的な繰り返し作業、手戻り作業および仕様変更の伝播の相互影響を踏まえて、シミュレーションモデルを提案し、作業の順位則について検討を行った。満行ら[29]は、情報システム開発プロジェクトに対して、手戻りを考慮し、遺伝的アルゴリズムを取り入れたシミュレーションモデルを開発し、同じく作業の優先ルールについて検討した。

2.3.3 組織構造に対する検討

組織構造をターゲットとしたシミュレーションモデルを紹介する。プロジェクトの組織構造を、シミュレーションベースで検討する代表的な研究として、2.2.2.1 でも触れた VDT (Virtual Design Team)[11]がある。VDT を基盤とし、様々の研究が展開された。Horii ら[30]は、IJV（International joint ventures）における異文化チームのメカニズムについて分析を行い、VDT モデルを構築し、チームのパフォーマンスを予測した。結果として、文化の差異により、組織スタイルが大きくチームのパフォーマンスに影響するという結論を述べている。Suzuki ら[31]

は、VDT の概念を取り入れ、PMT（Process Management Tool）を開発した。PMT を利用することによって、モデルベースでのビジネスの設計と管理が可能になる。満行ら[32]は、複数の複雑な設計プロジェクトに対して、各プロジェクトの異なる状況を考え、組織人員の配置ルールに関する意思決定を支援するシミュレーションモデルを開発した。なお、2.3.1 でも言及した B. R. Moser による Teamport[22]では、国際プロジェクトにおける組織構造を検討することも可能である。

2.3.4 作業現場の管理

シミュレーションモデルのもう 1 つの重要な用途は、作業現場の管理である。以下では、現場管理におけるシミュレーションツール、もしくはそれに関連する研究を述べる。

Enterprise Dynamics [33]は、INCONTROL Simulation Solutions 社により開発した作業現場管理用のシミュレーションツールであり、空港、運輸、倉庫、教育などの特性に応じたライブラリを複数搭載しており、20 年以上に渡り、運用されてきた。FlexSim[34]は、FlexSim Software Products 社により開発されたシミュレーションツールであり、ロジックビルディングツールが搭載され、高度なプログラミング知識のないユーザも、標準機能を活用し、複雑なシミュレーションモデルを構築することが可能である。

また、近年では、様々な生産管理設備が工場に導入されたため、これらの新しい設備を考慮したシミュレーションツールも多く開発されている。日比野[35]は、シミュレーション技術の1つである VM（Virtual Manufacturing）技術について紹介している。VM 技術を活用することによって、IoT（Internet of Things）設備などの生産管理設備を工場に導入した後の現場の振る舞いをシミュレートすることが可能となる。貝原[36]は、IoT 環境下のシミュレーションを含めた最新の研究をまとめ、神戸の地域産業であるラバー製スポーツシューズの生産を対象とし、実仮想融合型生産システムの取り組みについて紹介している。具体的なシミュレーションツールの例として、IOTSim[37]がある。IOTSim は、IoT 設備とクラウド環境に特化したシミュレーションツールであり、MapReduce モデルを利用したクラウド環境において、IoT 設備からの情報を取り入れたシミュレーションを行うことができる。また、Microsoft 社により開発した総合ソリューションツール Azure[38]においても、作業現場のシミュレーションモデルが搭載された。なお、IoT 設備などの生産設備を現場に導入する時の基盤を構築するために、森[39]は、設備情報をシミュレーションモデルに取り入れる際に使用する情報モデルの標準化を提案し、連動システム全体の構成と適用について検討した。

シミュレーションモデルの現場運用に対して評価を行う研究として、T. F. Edgar と E. N.

Pistikopoulos[40]は、化学工業のケースを用いて、知能生産システムの導入をデモンストレーションし、評価を行った。S. Masoud ら[41]は、農業作業現場の人員配置についてシミュレーションを行い、削減可能なコストに関する評価を行っている。

2.4 不確実性パラメータの設定

2.2 節と2.3 節で紹介した様々なシミュレーションモデルにおいて、パラメータの設定は難しい課題である場合が多い。特に、シミュレーションモデルの中で考慮する不確実性が多ければ多いほど、確率的なパラメータに関する情報が必要であるとされ、人手で設定することがより困難になる。T. R. Browning[2]の研究では、研究対象とするプロジェクトに関わる人員に向けて、アンケートを実施し、統計により、予想される工期、手戻り確率と手戻りの影響率などのパラメータを設定した。パラメータ設定の例として、M. W. Merkhofer[42]の研究では、グループ確率分布や連続確率の離散化などを含めた専門家による確率パラメータの設定プロセス、およびそれらの実用例についてレビューされている。レビューから得た知見として、確率パラメータの設定は非常に困難であり、通常、様々な情報が必要になるため、1つの値を決めるだけでも、1日以上の時間がかかることとなる。また、G. G. Shephard と C. W. Kirkwood[43]は、アナリストとマネージャーのコミュニケーションのケーススタディを用いて、インタビューベースで効率的に専門家の意見をまとめる手法を提案した。上記の研究を踏まえて、A. A. Yassine ら[44]は、タスク多様性に対する主観的評価、マッピングとキャリブレーション、検証の 3 つの段階を踏まえて、DSM を基盤としたシミュレーションモデルにおける手戻り確率の設定手法を提案した。

2.5 本研究の位置付け

2.4 節では、シミュレーションモデルにおける不確実性パラメータの設定手法を紹介した。既存の研究では、作業 1 つ 1 つに対して、手戻りの原因に対する調査、および一連のレーティング作業が必要である。従って、既存の研究は、ある程度実用性があると考えられるが、シミュレーションモデルの運用においては、依然として多くの労力が必要なプロセスになると考えられる。作業データの記録が一般的となっている現在においては、データを効率的に利用する手法の開発が可能となっていると考えられる。

本研究では、過去の実績データが存在するプロジェクトに対して、前述したような煩雑な作

業を軽減できる手法を提案する。実績データを円滑に利用し、不確実性パラメータを統計的に算出することによって、シミュレーションを行う。ただし、実績データの利用を想定しなかった、既存のシミュレーションモデルにおける不確実性パラメータを、実績データからそのまま算出することは、困難と考えられる。そこで本研究では、実績データからのパラメータ抽出を想定したシミュレーションモデルも、併せて提案する。その上で、提案したモデルを用いたシミュレーションによって、リソースの配置など、プロジェクトに関する意思決定を支援する。

以上を踏まえて、本研究の新規性は、以下の 2 点となる。

・実績データからの不確実性パラメータの抽出を想定した、遅延と手戻りを考慮したシミュレーションモデルの提案

・不確実性パラメータ抽出手法の提案

第3章 提案手法

3.1 はじめに

本章では、提案手法について説明する。第 1 章と第 2 章で言及した通り、本研究の提案手法は、シミュレーションモデルの提案、および実績データからのシミュレーションモデルの不確実性パラメータを抽出する手法の 2 つから構成される。提案手法の概要を述べた後、シミュレーションモデル、パラメータの抽出手法、抽出したパラメータを運用する時の例外処理、およびシミュレーションモデルを稼働させるためのシミュレータについて説明する。

3.2 提案手法の概要

提案手法の概要図を、Fig. 3-1 に示す。

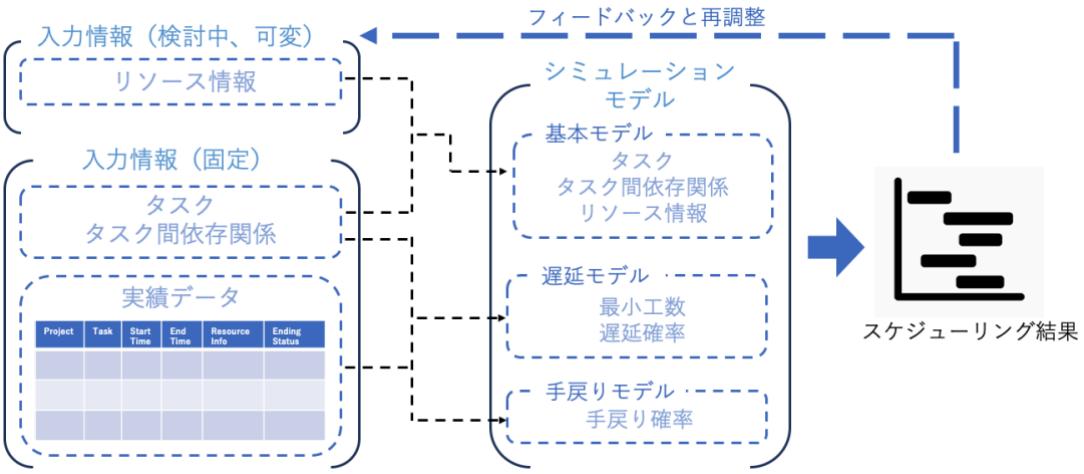

Fig. 3-1 提案手法の概要図

まず、提案手法全体の入力情報として、リソースの情報、プロジェクトを構成するタスクとタスク間依存関係の情報、及び過去の実績データがある。この中で、リソース情報は検討中のものとし、その他は固定な情報として扱う。シミュレーションモデルは、後述する基本モデル、遅延モデル、手戻りモデルから構成されている。基本モデルの情報は、入力情報であるリソースの情報と、タスクに関連する情報を使用し、遅延モデルと手戻りモデルの情報は、実績データから、タスクに関する情報を用いながら抽出するものとする。シミュレーションによるスケジューリングの結果は、プロジェクトにおいて検討中のリソース情報に対して、フィードバックを与える。提案手法を通して、実状況と需要に応じたリソース情報の再調整が可能となる。

3.3 提案するシミュレーションモデル

3.3.1 概要

シミュレーションモデルの概要図を Fig. 3-2 に示す。

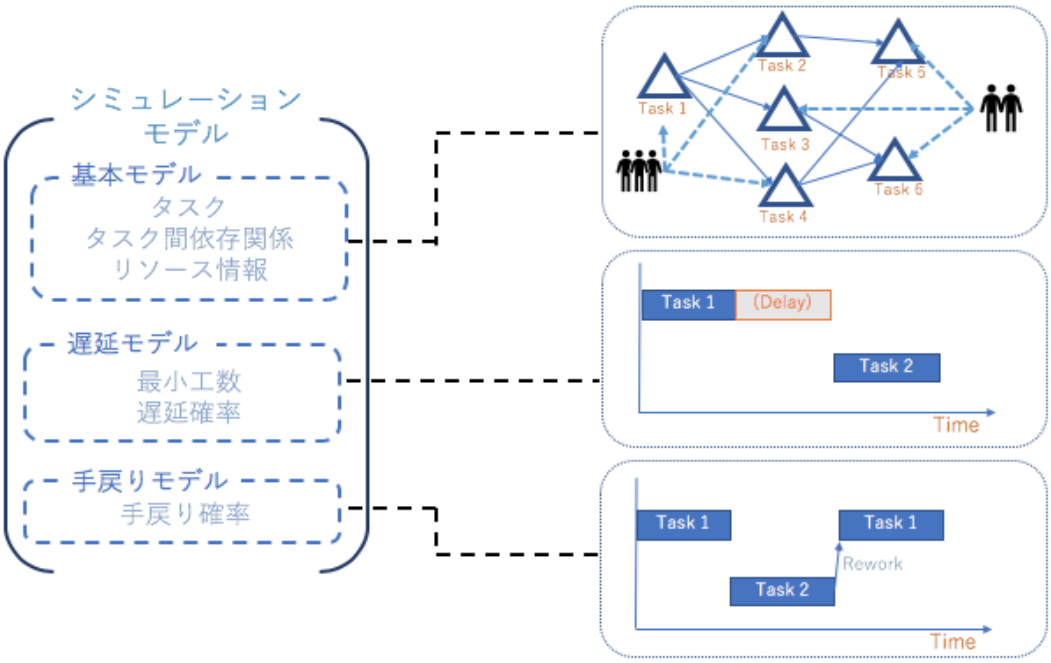

Fig. 3-2 シミュレーションモデルの概要図

シミュレーションモデルでは、前節で述べたように、基本モデル、遅延モデルと手戻りモデルの 3 つのパーツに分けられる。基本モデルでは、不確実性を含めないタスクと人員の基本情報をモデリングしている。また、遅延モデルと手戻りモデルは、それぞれ、タスクの基本工数をベースラインとする遅延、及び繰り返し作業を代表する手戻り作業を、プロジェクトの不確実性要素としてモデリングしている。本節では、モデルの構成から、不確実性に関する仮説とシミュレーションの手順まで、詳細に説明する。

3.3.2 シミュレーションモデルの構成

3.3.2.1 基本モデル

基本モデルでは、細分されたプロジェクトの各手順と人員配置といった基本的な構造をモデリングする。先行研究である[24][25]とは、類似するモデルとなっている。

プロジェクト全体を複数のタスクに分け、タスクの間には、一定の依存関係が存在するものとする。この概念は、DSM を中心としたモデル[12][13][20]と同様であるため、本研究でも、DSM を用いて、タスクとタスク間の依存関係を表現する。実作業におけるタスク間の依存関係は、明確に定義できないものが多く、依存関係の種類も多様であるが、本研究では、提案手法全体の複雑さを考慮した上で、古典的かつ最も一般的な Finish-to-Start のみを使用する。以下では、簡単な例を用いて、タスクとタスク間依存関係のモデルを説明する。

1 つの家具を設計し、製造するプロセスを1つのプロジェクトとして考える。このプロジェクトは、コンセプト設計、全体設計、部品設計、原材料購入、部品購入、部品製造、組み立て、塗装の 8 つのタスクに分けることができると仮定する。リソースが十分であることを前提にしたワークフローを Fig. 3-3 に示す。

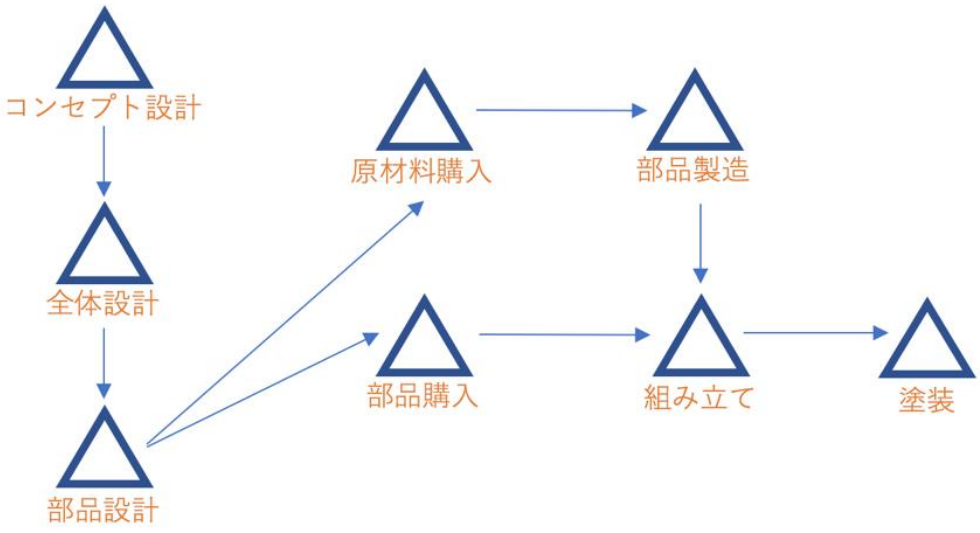

Fig. 3-3 基本モデルの例（家具の設計と製造プロジェクト）

Fig. 3-3 では、各矢印は、Finish-to-Start の依存関係を示している。矢印後のタスクは、矢印前のタスクが完了するまで実行できない。Fig. 3-3 のプロジェクトを DSM で表すと、Table

3-1 のようなマトリクスとなる。

Table 3-1 では、バイナリの値で依存関係を表現している。DSM の一般的なルールとして、依存関係は対角線以下の各要素で表され、1である要素は、横の ID を持つタスクが、縦の ID を持つタスクからの情報やものが必要であることを示す。補足として、DSM では、行列の各要素を、バイナリの値ではなく、数値を配置することによって、様々な情報を表せるが、本研究では、タスクとタスク間の依存関係のみを表現している。

Table 3-1 家具の設計と製造プロジェクトの DSM

タスク名	ID	1	2	3	4	5	6	7	8
コンセプト設計	1		0	0	0	0	0	0	0
全体設計	2	1		0	0	0	0	0	0
部品設計	3	0	1		0	0	0	0	0
原材料購入	4	0	0	1		0	0	0	0
部品購入	5	0	0	1	0		0	0	0
部品製造	6	0	0	0	1	0		0	0
組み立て	7	0	0	0	0	1	1		0
塗装	8	0	0	0	0	0	0	1	

各タスクt_i、及びそれらが依存しているタスクの集合IT_iを、式(3.1)と(3.2)のように定式化するiとNはそれぞれ、タスクの ID とプロジェクトにおけるタスクの数である。

$$t_i, i = 1,2, \dots, N \tag{3.1}$$

$$IT_i, i = 1,2, \dots, N \tag{3.2}$$

また、本研究のシミュレーションモデルは、資源制約つきスケジューリング問題として扱われるため、リソースを定義する必要がある。ここで、T. Mitsuyuki ら[24]が使用したリソースモデルを採用する。以下では、引き続き、家具設計と製造プロジェクトの例を用いて、説明する。

各タスクが必要とするリソースは、Table 3-2 であるとする。

Table 3-2 家具の設計と製造プロジェクトの必要リソース

ID	タスク名	必要リソース
1	コンセプト設計	設計人員
2	全体設計	設計人員、パソコン
3	部品設計	設計人員、パソコン
4	原材料購入	調達人員、トラック
5	部品購入	調達人員、トラック
6	部品製造	製造人員、製造機械
7	組み立て	組み立て人員、組み立て道具
8	塗装	塗装人員、塗装工具

一般的に、大規模プロジェクトでは、リソースの専門性が高い場合が多いため、1 つのタスクが必要とする人員と機械などを、抽象化されたリソースとしてモデリングする。家具の例では、Table 3-3 のようなリソースチームがあるとする。リソースに関するタスクを、リソースのスキルとして記述する。Table 3-4 のように、リソース 1 つに対して、タスク毎に、スキルが付与される。

Table 3-3 家具の設計と製造プロジェクトのリソースチーム

ID	リソース	チーム構成
1	設計チーム	設計人員 1 名、パソコン 1 台
2	調達チーム 1	調達人員 1 名、トラック 1 台
3	調達チーム 2	調達人員 2 名、トラック 2 台
4	製造チーム 1	製造人員 1 名、製造機械 1 台
5	製造チーム 2	製造人員 2 名、製造機械 1 台
6	組み立てチーム	組み立て人員 1 名、組み立て道具 1 セット
7	塗装チーム	塗装人員 1 名、塗装工具 1 セット

Table 3-4 家具の設計と製造プロジェクトにおけるリソースのスキル表

ID / スキル	1	2	3	4	5	6	7
コンセプト設計	1	0	0	0	0	0	0
全体設計	1	0	0	0	0	0	0
部品設計	1	0	0	0	0	0	0
原材料購入	0	1	2	0	0	0	0
部品購入	0	1	2	0	0	0	0
部品製造	0	0	0	1	2	0	0
組み立て	0	0	0	0	0	1	0
塗装	0	0	0	0	0	0	1
（工数/時間）							

各リソースr_i、及びリソースのスキルを表すベクトルrs_iは、式(3.3)と(3.4)のように定式化する。ここで、Mはリソースの数であり、$s_{i1}, s_{i2}, \dots, s_{iN}$は、リソース$r_i$のタスク$t_1, t_2, \dots, t_N$に対するスキルの値である。

$$r_i, i = 1,2, \dots, M \tag{3.3}$$

$$rs_i = (s_{i1}, s_{i2}, \dots, s_{iN}), i = 1,2, \dots, M \tag{3.4}$$

3.3.2.2 手戻り作業に関する仮定

遅延モデルと手戻りモデルを説明する前提として、本研究におけるプロジェクトの手戻り作業に関する仮定を、引き続き家具の設計と製造プロジェクトを例にして説明する。

リソースを考慮しない場合、依存関係を遵守する前提で、各タスクの実行順は、Fig. 3-4 のようになる。ここでは、部品設計の実行時に、コンセプト設計あるいは全体設計での重大な設計ミスを発見し、再度、上流設計の実行が余儀なくされたと仮定する。初めての部品設計時に、全体設計における手戻りが必要となり、全体設計からやり直したとする。この場合の各タスクの実行順と実行回数を Fig. 3-5 に示す。

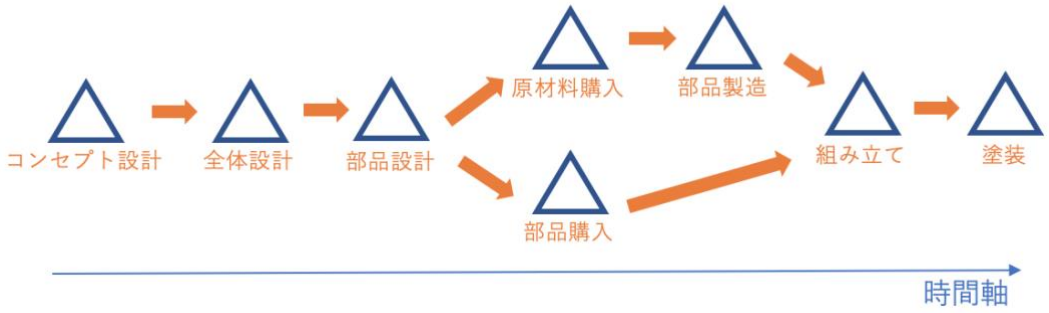

Fig. 3-4 手戻り作業に関する仮定：説明図 1

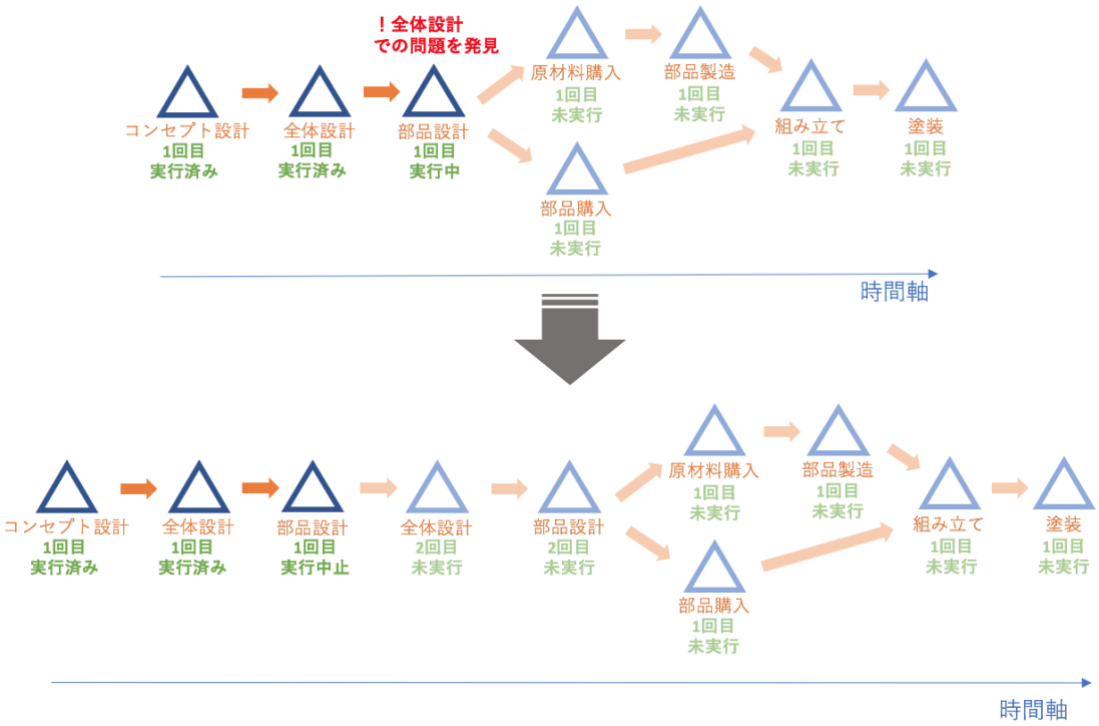

Fig. 3-5 手戻り作業に関する仮定：説明図 2

さらに、1 回目の原材料購入と 1 回目の部品購入が同時に進行している途中に、部品設計における手戻りが必要となった場合を考える。その場合、原材料購入も部品設計からの情報が必要であるため、部品購入自体だけでなく、原材料購入も再度実行することになる。この場合のタスクの実行順と実行回数を、Fig. 3-6 に示す。

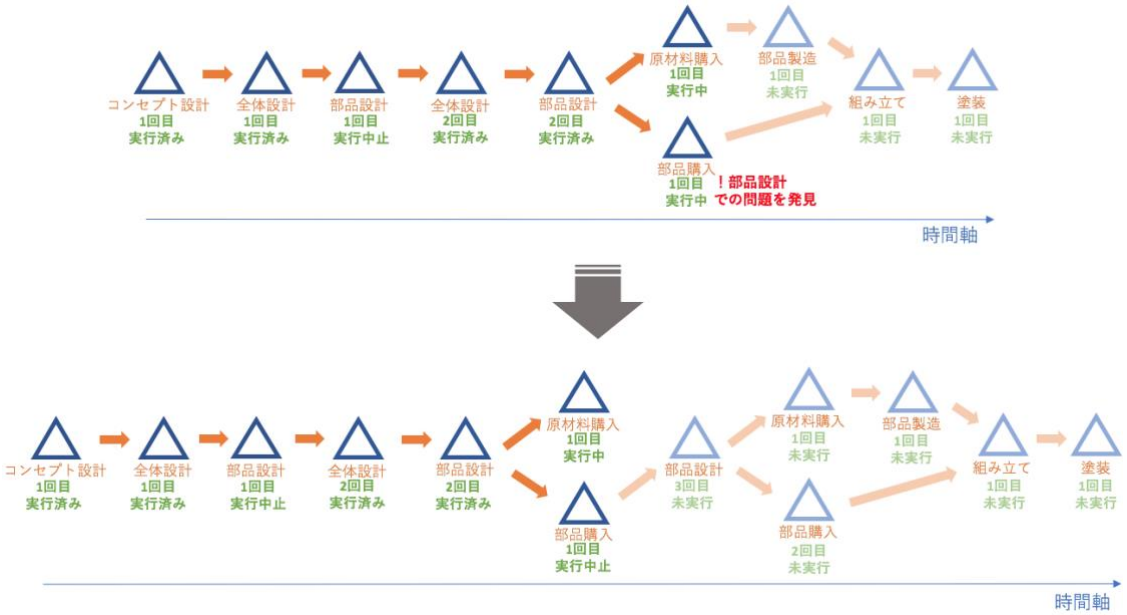

Fig. 3-6 手戻り作業に関する仮定：説明図 3

本研究では、上記のような手戻りを仮定し、各タスクに対して、タスク自体だけでなく、タスクの実行回数毎に変化する属性値を与える。詳細については、後述する。

3.3.2.3 遅延モデル

遅延モデルは、タスクにおける最小の工数、及び最小の工数に基づく一連の遅延確率から構成される。

最小の工数とは、特定実行回のタスクにおける、完了するまでに最低限必要とする工数であり、式(3.5)のように示す。ここで、$mwa_{i,o}$は、タスクt_iのo回目の実行における最小の工数である。また、不確実性の要素として、遅延モデルでは、式(3.6)のように遅延確率が定義される。ここで、$dp_{i,o,wa}$は、タスクt_iのo回目の実行において、waの工数が追加された確率である。waの値は正整数とし、$dp_{i,o,wa}$の値は、0 から1の実数とする。タスクt_iのo回目の実行に対して、このような確率$dp_{i,o,wa}$の値を、複数与えることは可能であるが、確率の総和は1とする。ただし、$wa = 0$は、遅延が発生しないことを意味する。

$$mwa_{i,o} > 0, i = 1,2,\dots,N, o \in \mathbb{N} \tag{3.5}$$

$$dp_{i,o,wa} \in (0,1], i = 1,2,\dots,N, o = 1,2,\dots,O, wa \in \mathbb{N} \tag{3.6}$$

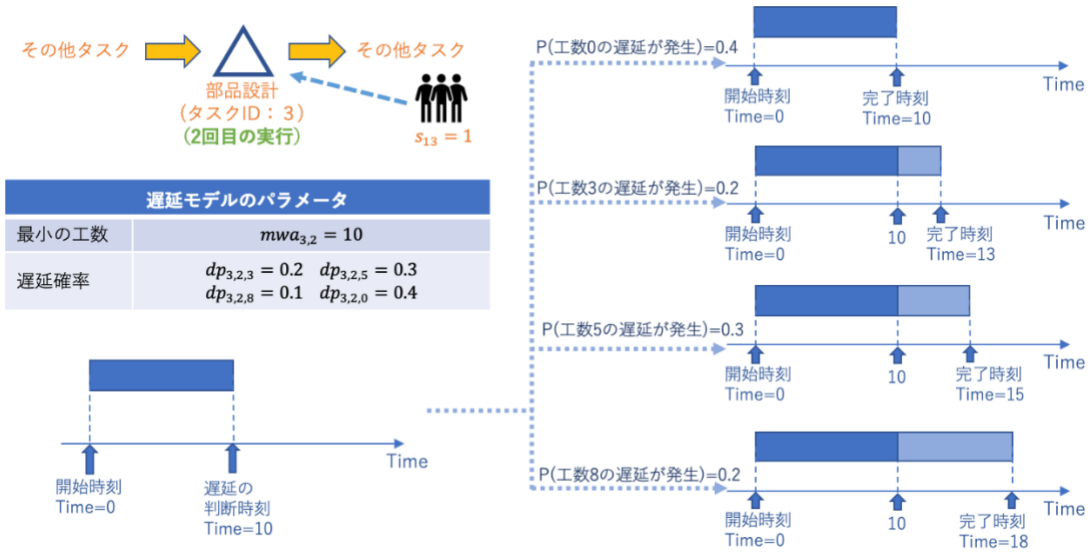

Fig. 3-7 遅延モデルの説明例

シミュレーション上の挙動を、Fig. 3-7 の例を用いて説明する。部品設計のタスクをt_3とし、2 回目の実行中に、ID が 1 のリソースr_1がアロケートされているとする。また、該当タスクの遅延モデルのパラメータは、左側の表のように与えられ、r_1のt_3に対するスキル値s_{13}は1である。この時、最小の工数を超えるまでに所要する時間は、$10 \div 1 = 10$である。最小の工数を超えた時点で、遅延の有無を判断するために、0 から 1 の区間上で一様分布に基づく乱数が生成される。遅延モデルのパラメータに従うと、遅延が発生しない確率が 0.4、工数 3 の遅延が発生する確率が 0.2、工数 5 の遅延が発生する確率が 0.3、工数 8 の遅延が発生する確率が 0.1 である。乱数の値が 0 から 0.4 の時は、遅延なしとし、t_3の 2 回目の実行を終了させる。乱数の値が 0.4 から 0.6 の場合は、工数 3 の遅延が発生するとし、残り工数を増やし、作業時間も遅延によって延長する。工数 5 と工数 8 の場合の遅延も同様に発生するため、遅延のパターンとそれに対応する確率は、右側のガントチャートのように表される。

提案する遅延モデルは、最小の工数は把握できるが、一度最小工数まで辿り着かないと、遅延が発生するかどうか、どのような遅延が発生するかが明確でない状況を想定し、モデリングしたものである。また、実運用では、「工数」の単位が、リソースのスキルの単位とスケジューリングが必要とする単位の乗積であり、具体的なプロジェクトの背景と需要に応じて、柔軟に選択できるものとする。

3.3.2.4 手戻りモデル

手戻りモデルは、一連の手戻り確率から構成される。手戻り確率とは、タスクの実行中に、他のタスクから手戻りする確率である。まず、タスクの時刻tにおける実行回数を式(3.7)で定義し、対応する進捗率を、式(3.8)で定義する。ここで、$pro_i(t)$は、時刻tにおけるタスクt_iの進捗率であり、$awa_i(t)$は、同じく時刻tにおける、現実行回数$o = o_i(t)$での実行された工数である。進捗率は 0.1 を単位とする。実際工数と、該当タスクの最小の工数の比を持って、進捗率とする。ただし、遅延が発生した場合、進捗率が 1 を超える。次に、手戻り確率を式(3.9)で示す。rp_{i_1,oc,pro,i_2}は、タスクt_{i_1}のoc回目の実行中に、進捗率がproである時に、タスクt_{i_2}へと手戻りする確率である。

$$o_i(t) \in \mathbb{N} \tag{3.7}$$

$$pro_i(t) = floor(10 \times awa_i(t)/mwa_{i,oc})/10 \tag{3.8}$$

$$rp_{i_1,oc,pro,i_2} > 0$$
$$i_1 = 1,2,\dots,N, i_2 = 1,2,\dots,i_1 - 1, i_1 + 1,\dots,N, o \in \mathbb{N} \tag{3.9}$$

シミュレーション上の挙動を、Fig. 3-8 の例を用いて説明する。家具の設計と製造プロジェクトにおいて、コンセプト設計、全体設計及び部品設計の3つのタスクを考える。これらのタスクは、依存関係によって、並行作業が起きず、線形的な実行順となっている。割り振られたリソースは 1 チームのみとし、各設計タスクに対して、各々1 のスキルを持つ。この設定に従うと、左下のようなガントチャートになる。青の部分は通常の作業であり、オレンジの部分は、手戻りによる作業である。

開始時刻を 0 とし、時刻が 36 である時、部品設計の現実行回数（1 回目）における実行済みの工数は 11 であり、最小工数の 18 から、進捗率は、$pro_3(36) = floor(10 \times 11 \div 18) \div 10 = 0.6$となる。手戻りモデルパラメータの$rp_{3,1,0.6,1}$に対応すると、この時刻では、0.2 の確率で、タスク ID が 1 であるコンセプト設計へと手戻りする。遅延モデルにおける遅延発生の判断と同じく、0 から 1 の区間上で、一様分布の乱数が1つ生成される。この乱数の値が、0.2 より小さい場合、手戻りが発生する。

時刻 36 における手戻り発生時の状態は、右上のガントチャートで示す。まず、手戻り先であるコンセプト設計のパラメータは、2 回目実行時のパラメータとなる。ここでは、2 回目実行時の最小の工数が 7 となる。次に、コンセプト設計の進捗率と実行済みの工数を 0 とする。最後に、コンセプト設計に依存する諸々のタスクにも、同じ処理を行う。つまり、手戻りにおけ

る一連の処理の後に、各設計タスクの最小の工数、進捗率と実行済みの工数は、式(3.10)から式(3.12)のようになる。

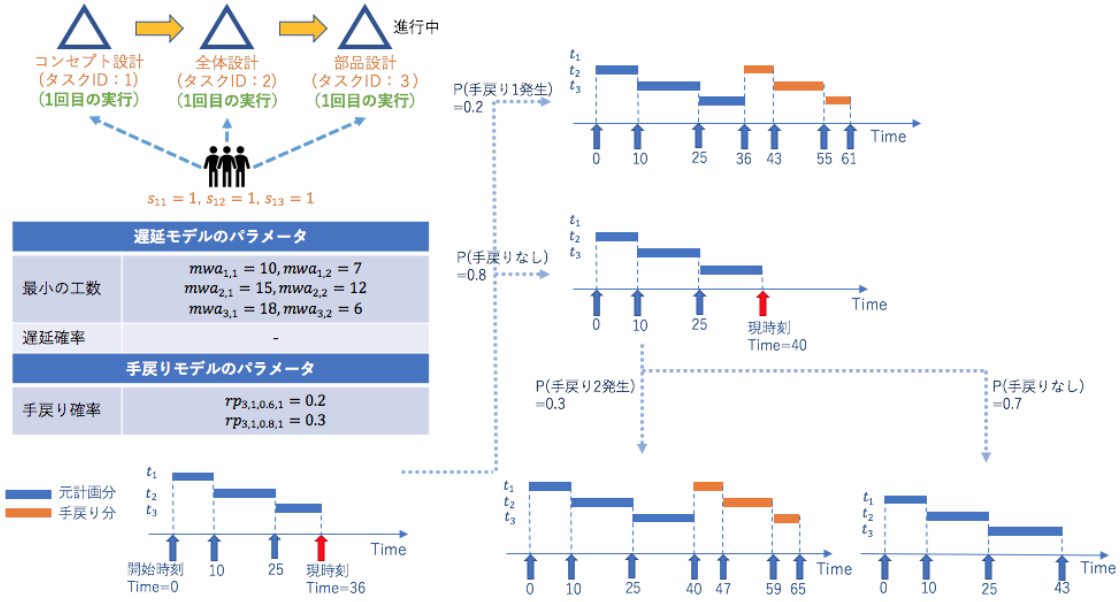

Fig. 3-8 手戻りモデルの説明例

$$mwa_{1,2}=7, awa_1(36)=0, pro_1(36)=0 \tag{3.10}$$

$$mwa_{2,2}=12, awa_2(36)=0, pro_2(36)=0 \tag{3.11}$$

$$mwa_{3,2}=6, awa_3(36)=0, pro_3(36)=0 \tag{3.12}$$

乱数の値が 0.2 より大きい場合、手戻りは発生せず、そのままタスクの実行を続行する。Fig. 3-8 の右下のガントチャートは、その後の進捗率に対応する手戻り確率のパラメータが存在する時の状況を示している。

また、タスクに手戻りが発生しないまま、実行工数がタスクの最小工数、あるいは遅延時の遅延工数と最小工数の和に達した時、作業が終了し、下流のタスクの実行に移行する。

提案する手戻りモデルは、Cho ら[20]が提案した繰り返し作業のモデルと類似しているが、作業の進捗率に合わせて、手戻りが発生するか否かを判断する点で異なるモデルとなっている。また、本研究では、上流タスクからの手戻りのみを想定し、前述したのように、タスク間の依存関係に対しても、厳密に従うものとする。すなわち、上流タスクの再度実行は、必ず実行中もしくは実行済みの下流タスクの調整を引き起こすものとする。ただし、下流タスクのう

ち、まだ進行中のタスクに対しては、実行回数を更新しないまま、進捗率を 0 に戻す。

3.3.2.5 シミュレーションの流れ

シミュレーションの大まかな流れを Fig. 3-9 に示す。

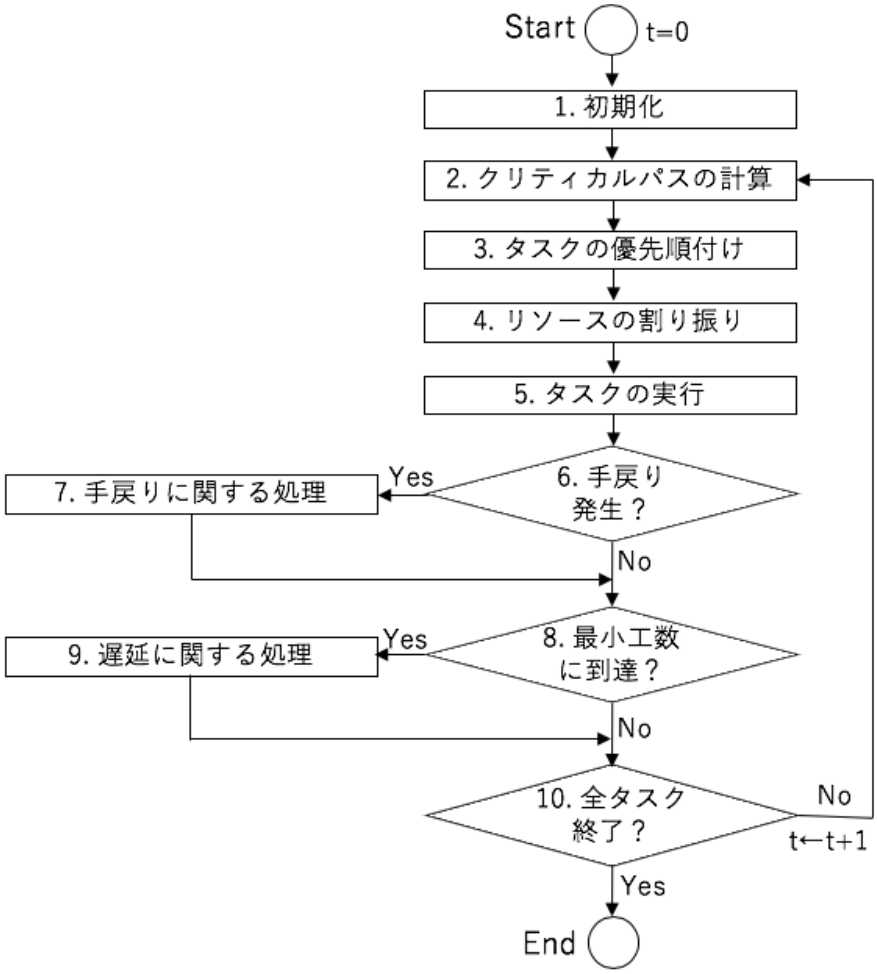

Fig. 3-9 シミュレーションのフローチャート

前述した通り、本研究では、離散イベントシミュレーションを用いる。時刻毎にモデルの状態を更新し、シミュレーションの終了判定がされるまで、一定の手順を繰り返す仕組みである。シミュレーションモデル上の各要素を、Table 3-5 にまとめる。ここで、各要素を、タスクに関するパラメータ、リソースに関するパラメータ、及び固定的な設定値と、時刻毎に変化する変化値の 4 つの枠に分類する。

Table 3-5 シミュレーションモデル上の要素

	タスクに関する集合と変数	
	要素名	数式
固定値	全タスクの集合	$AT = \{t_1, t_2, \dots, t_N\}$
	タスク ID	$i \in \{1,2, \dots, N\}$
	タスクt_iの先行タスクの集合	$IT_i \subset AT$
	タスクt_iに依存するタスクの集合	$OT_i \subset AT$
	タスクt_iの最小工数	$mwa_{i,o}, o \in \mathbb{Z}$
	タスクt_iの遅延確率（複数）	$dp_{i,o,wa}, o \in \mathbb{Z}, wa \in \mathbb{Z}$
	タスクt_iの手戻り確率（複数）	$rp_{i,o,pro,j}, o \in \mathbb{Z}, pro \geq 0,$ $j \in \{1,2, \dots, i-1, i+1, \dots, N\}$
変化値	実行中のタスクの集合	$WT(t) \subseteq AT$
	遅延中のタスクの集合	$DT(t) \subseteq WT(t)$
	完了した全タスクの集合	$CT(t) \subseteq AT$
	未実行のタスクの集合	$NT(t) \subseteq AT$
	タスクt_iが配分されたリソースの集合	$R_i(t) \subseteq AR$
	タスクt_iの実行された工数	$awa_i(t) \geq 0$
	タスクt_iの実行回数	$o_i(t) \in \mathbb{Z}$
	タスクt_iの残り工数	$rwa_i(t) \geq 0$
	タスクt_iの進捗率	$pro_i(t) \geq 0$
	タスクt_iの最早開始時間	$es_i(t)$
	タスクt_iの最遅開始時間	$ls_i(t)$
	リソースに関する集合と変数	
	要素名	数式
固定値	全リソースの集合	$AR = \{r_1, r_2, \dots, r_M\}$
	リソースr_iのスキル	$rs_i = (s_{i1}, s_{i2}, \dots, s_{iN})$
変化値	作業中のリソースの集合	$WR(t) \subseteq AR$
	フリーのリソースの集合	$FR(t) \subseteq AR$

定式化されたシミュレーションの各挙動の詳細は、以下に述べる。

1. 初期化:

この手順では、時刻tを 0 にし、タスクとリソースの各固定値が読み込まれ、式(3.13)から式(3.18)のように、タスクとリソースの状態が初期化される。

$$WR(0) \leftarrow \emptyset, FR(0) \leftarrow AR \tag{3.13}$$

$$WT(0) \leftarrow \emptyset, CT(0) \leftarrow \emptyset, NT(0) \leftarrow AT \tag{3.14}$$

$$R_i(0) \leftarrow \emptyset, i \in \{1,2,\dots,N\} \tag{3.15}$$

$$awa_i(0) \leftarrow 0, i \in \{1,2,\dots,N\} \tag{3.16}$$

$$o_i(0) \leftarrow 1, i \in \{1,2,\dots,N\} \tag{3.17}$$

$$rwa_i(0) \leftarrow mwa_{i,1}, i \in \{1,2,\dots,N\} \tag{3.18}$$

2. クリティカルパスの計算:

完了していないタスク($t_i \in WT(t) \cup NT(t)$)に対して、残り工数をタスクの工数とし、クリティカルパスを計算する。詳細な計算手順は、[8]を参照する。結果として、タスクの最早開始時間$es_i(t)$と最遅開始時間$ls_i(t)$が算出される。

3. タスクの優先順付け:

完了していないタスク($t_i \in WT(t) \cup NT(t)$)に対して、TSLACK の優先度ルール[25]に従い、ソートを行う。TSLACK とは、クリティカルパス上の近いタスクを優先するルールであり、最遅開始時間と最早開始時間の差($ls_i(t) - es_i(t)$)の昇順で、タスクがソートされる。

4. リソースの割り振り:

フリーのリソース($r_i \in FR(t)$)を、タスクへ割り振る。先頭のタスクから、1つずつ、式(3.19)と式(3.20)のように、リソースが配分される。ここで、処理に当たるタスクをt_kと記述する。

$$R_k(t) \leftarrow \{r_i \in FR(t) | s_{ik} > 0\} \tag{3.19}$$

$$WR(t) \leftarrow WR(t) \cup R_k(t), FR(t) \leftarrow AR - WR(t) \tag{3.20}$$

5. タスクの実行

タスクを実行する条件は、式(3.21)に示す通り、全ての先行タスクが完了し、リソースが割り振られていることである。$WT(t)$の更新に伴い、$NT(t)$も、式(3.22)のように更新される。

$$WT(t) \leftarrow \{t_i \in AT - CT(t) | DT_i \subset CT(t) \; and \; R_i(t) \neq \emptyset\} \tag{3.21}$$

$$NT(t) = AT - CT(t) - WT(t) \tag{3.22}$$

実行中のタスク($t_i \in WT(t)$)に対して、残り工数から、タスクを実行しているリソースの該当タスクに対するスキル値の総和を引き、同じ値を実行された工数に加える。詳細を式(3.23)と式(3.24)に示す。

$$rwa_i(t) \leftarrow rwa_i(t) - \sum_{r_k \in R_i(t)} s_{ki} \tag{3.23}$$

$$awa_i(t) \leftarrow awa_i(t) + \sum_{r_k \in R_i(t)} s_{ki} \tag{3.24}$$

6. 手戻りに関する判断

実行中のタスク($t_i \in WT(t)$)に対して、3.3.2.4 で述べたように、進捗率を計算し、手戻りの発生を判断する。

7. 手戻りに関する処理

手戻りが発生したタスクをt_iと記述し、手戻り先となるタスクをt_jと記述する。t_iとt_jに対する処理を、式(3.25)から式(3.28)に示す。

$$o_i(t) \leftarrow o_i(t) + 1 \quad o_j(t) \leftarrow o_j(t) + 1 \tag{3.25}$$

$$rwa_i(t) \leftarrow mwa_{i,o_i(t)} \quad rwa_j(t) \leftarrow mwa_{j,o_j(t)} \tag{3.26}$$

$$awa_i(t) \leftarrow 0 \quad awa_j(t) \leftarrow 0 \tag{3.27}$$

$$NT(t) \leftarrow NT(t) \cup \{t_i, t_j\} \tag{3.28}$$

ここで、$t_i \in WT(t)$であるため、また式(3.29)から式(3.31)の処理を順に行う。また、$t_j \in WT(t)$の場合では、t_jに対してt_iと同様な処理を行う。

$$WT(t) \leftarrow WT(t) - \{t_i\} \tag{3.29}$$

$$FR(t) \leftarrow FR(t) \cup R_i(t), WR(t) \leftarrow WR(t) - R_i(t) \tag{3.30}$$

$$R_i(t) \leftarrow \emptyset \tag{3.31}$$

$t_j \in CT(t)$の場合、式(3.29)、式(3.30)と式(3.31)の代わりに、式(3.32)の処理を行う。

$$CT(t) \leftarrow CT(t) - \{t_j\} \tag{3.32}$$

その後、t_jに依存しているタスクの中で、すでに完了しているタスク($t_k \in OT_j \cap CT(t)$)に対しては、t_jと同様の処理を行う。また、t_jに依存しかつ進行中のタスク($t_n \in OT_j \cap WT(t)$)に対しては、式(3.25)の処理を行えず、その他の処理を同様に行う。

8. 遅延に関する判断

実行中に一度も遅延が発生したことがないタスク($t_i \in WT(t) - DT(t)$)に対して、$rwa_i(t) \leq 0$、つまり進捗率が 1 に達した時に、3.3.2.3 で述べた遅延の判断手順に従い、遅延工数waを読み込む。

9. 遅延に関する処理

$wa = 0$の場合、t_iを完了とし、式(3.33)から式(3.35)の処理を順に行う。

$$CT(t) \leftarrow CT(t) \cup \{t_i\}, WT(t) \leftarrow WT(t) - \{t_i\} \tag{3.33}$$

$$FR(t) \leftarrow FR(t) \cup R_i(t), WR(t) \leftarrow WR(t) - R_i(t) \tag{3.34}$$

$$R_i(t) \leftarrow \emptyset \tag{3.35}$$

$wa > 0$の場合、式(3.36)と式(3.37)の処理を行う。

$$DT(t) \leftarrow DT(t) \cup \{t_i\} \tag{3.36}$$

$$rwa_i(t) \leftarrow rwa_i(t) + wa \tag{3.37}$$

10. シミュレーション終了の判断:

$CT(t) = AT$の場合、シミュレーションを終了し、ガントチャートを出力する。終了していないタスクがある場合、 時刻t ← t + 1し、2. クリティカルパスの計算から、シミュレーションを続行する。

3.4 不確実性パラメータの抽出

3.4.1 概要

本節では、不確実性パラメータの抽出で使用するアルゴリズムとプログラムについて、詳細に説明する。抽出手法全体の概要図を Fig. 3-10 に示す。

実績データが読み込まれた後、まず、各プロジェクトに必要とする付加情報（DSM）と合わせて、処理を行う。次に、各プロジェクトにおける処理済みのデータをもとに、遅延モデルと手戻りモデルの要素を集計するためのリストを作成する。最後に、一定のルールに従い、集計を行う。これらの集計結果を不確実性パラメータとする。

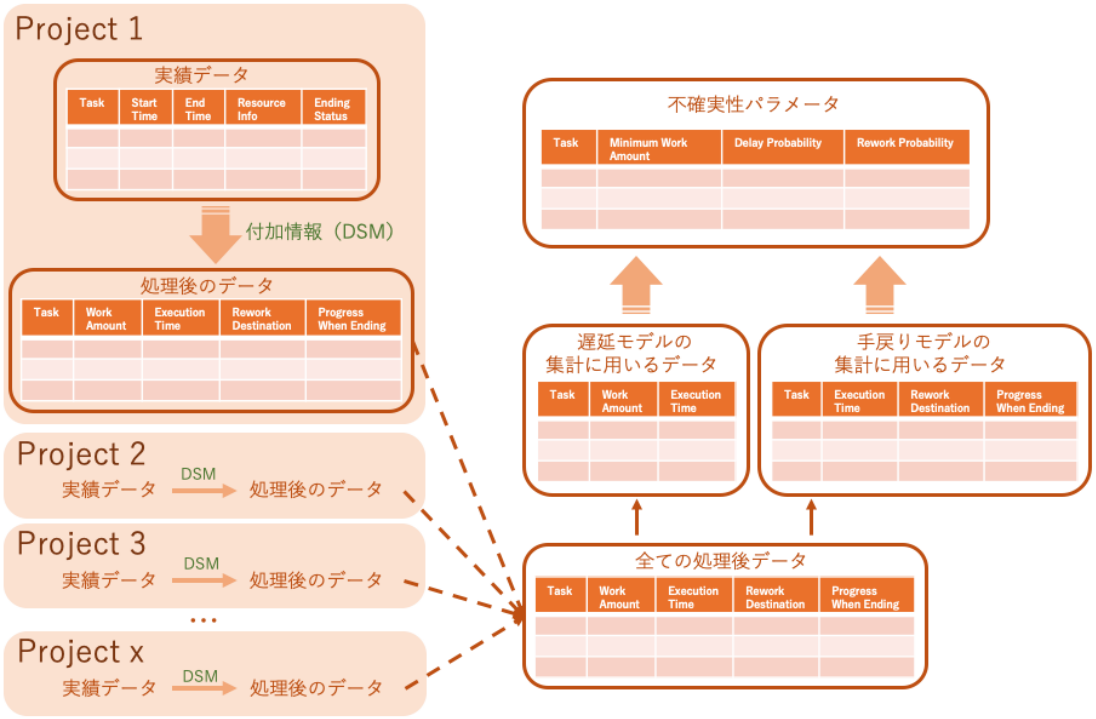

Fig. 3-10 抽出手法の概要

また、プログラム上で、主な計算や集計を行うクラスとデータ構造に用いるクラスの概要を Table 3-6 に示す。

Table 3-6 パラメータ抽出プログラムの概要

	クラス名	概要とデータ構造
計算と集計	Project	プロジェクト毎に行う集計と計算（実行回数のカウント、手戻り先の集計） - List<Task> taskList：タスクの情報 - List<Data> dataList：該当プロジェクトに関する実績データ
	Extraction	不確実性パラメータの集計 - List<Project> projectList：実績データ中の全てのプロジェクト - List<Task> taskList：タスクの情報 - List<Data> dataList：全ての実績データ
データ構造	Data	- (String) project：実績データにおけるプロジェクト名 - (String) task：実績データにおけるタスク名 - (Integer) st：開始時間 - (Integer) et：終了時間 - (Double) resourceCapacity：リソースのスキル - (String) status：終了時のステータス（手戻り発生しなかった場合 : Normal もしくは Suspend ; 手戻りは発生した場合、手戻り先のタスク名 ; 複数の手戻り先がある場合、";"で分割） - (Integer) oc：プロジェクトにおける実行回数 - (String) reworkTask：集計時に用いる、手戻り発生時の手戻り先 - (Double) progressWhenEnd：終了時の進捗率
	Task	- (String) name：タスク名 - (Map<Integer, Double>) mwa：最小の工数 - (List<Task>) inputTaskList：直接依存しているタスク - (Delay) delay：遅延確率 - (Rework) rework：手戻り確率
	Rework	- (List<Integer>) oc：実行回数 - (List<Double>) progress：進捗率 - (List<Double>) probability：手戻りが発生する確率 - (List<Task>) des：手戻り先のタスク
	Delay	- (List<Integer>) oc：実行回数 - (List<Double>) probability：遅延が発生する確率 - (List<Integer>) wa：遅延工数

3.4.2 必要とする実績データ

本研究で扱う実績データには、データ 1 件の内、プロジェクト名、タスクあるいはアクティビティ(以下:タスク)名、開始時の日時、終了時の日時、該当タスクに配分されたリソース情報、終了時もしくは中断時の情報の 6 つの項目がある。Table 3-7 に、1つの仮想プロジェクトの実績データを、例として示す。

Table 3-7 実績データの例：プロジェクト p1

ID	Project	Task	Start Time	End Time	Resource Info	Ending Status
001	p1	コンセプト設計	150302_09(0)	150303_12(11)	5 工数/h	Normal
002	p1	全体設計	150303_10(9)	150305_16(30)	5 工数/h	Normal
003	p1	部品設計	150306_09(32)	150310_17(55)	5 工数/h	Normal
004	p1	原材料購入	150311_09(55)	150311_11(57)	10 工数/h	Rework:コンセプト設計
005	p1	部品購入	150311_09(55)	150311_12(58)	7 工数/h	Suspend
006	p1	全体設計	150311_10(56)	150311_18(63)	5 工数/h	Normal
007	p1	部品設計	150312_09(63)	150312_12(66)	7 工数/h	Normal
008	p1	原材料購入	150312_13(66)	150312_18(71)	7 工数/h	Normal
009	p1	部品購入	150312_13(66)	150312_18(71)	5 工数/h	Normal
010	p1	部品製造	150312_14(67)	150318_16(101)	20 工数/h	Rework:部品設計
011	p1	部品設計	150319_09(103)	150319_12(106)	5 工数/h	Normal
012	p1	原材料購入	150319_13(106)	150319_18(111)	2 工数/h	Normal
013	p1	部品購入	150319_13(106)	150320_18(119)	2 工数/h	Normal
014	p1	部品製造	150323_09(119)	150324_12(131)	20 工数/h	Normal
015	p1	組み立て	150324_13(131)	150327_17(160)	8 工数/h	Normal
016	p1	塗装	150324_15(133)	150327_12(156)	3 工数/h	Normal
017	p1	部品製造	150326_11(146)	150326_18(153)	15 工数/h	Normal
018	p1	塗装	150330_09(161)	150331_12(172)	3 工数/h	Normal

ここでは、開始時の日時と終了時の日時が詳細に記録されているが、工数を計算するために、労働時間数だけをカウントし、時刻を括弧内に示す。リソース情報とは、リソースのスキルである。工数の算出に必要であるため、事前にリソースを抽象化し、スキルを決める必要がある。また、終了時もしくは中断時の情報とは、1 件の実績データに関する情報であり、以下の Normal、Rework、Suspend の 3 種がある:

- Normal: データ記録時に、該当タスクが異常なく終了したことを意味する。
- Rework: 手戻りが発生したため、作業が中止されたことを意味する。ただし、手戻り先となったタスクも同時に記録される。手戻り先は、複数あることも可能である。
- Suspend: 手戻り以外の原因（依存関係、リソースの不足など）による作業の中止を意味する。

3.4.3 必要とする付加情報

各パラメータを計算するために、タスク間の依存情報に関する情報を入力する必要がある。依存関係の情報は、3.3.2.1 で述べた基本モデルを使用し、DSM で記述する。例えば、家具の設計と製造の例に対応する DSM は、Table 3-1 に示す通りである。

本研究は、規模と実行手順が同様なプロジェクトを、異なるリソース状況で、新しく実行する前に、工期を見積もる手法を提案するため、過去の複数のプロジェクトに関するタスク間依存関係の情報も、同様となっている。詳細な議論は、考察で述べる。

3.4.4 前処理

前処理では、計算と集計を行う前に、シミュレーションモデルの各仮定を踏まえ、実績データに対して処理を行う。3.4.4.1 と3.4.4.2 では、それぞれ、タスクの依存関係に関する前処理、および終了時ステータスが「Suspend」である実績データに対する処理を、詳細に記述する。

3.4.4.1 依存関係に関する前処理

現場の状況によっては、定義された依存関係に厳密に従わない実績データが記録される場合がある。前処理の段階では、実績データに対して、適切な処理を行うことによって、依存関係と一致するデータにする。

Table 3-7 にある ID が 001 のデータと、ID が 002 の番号が、依存関係に反した一例である。この 2 件のデータとガントチャート、および前処理を行った後のデータとガントチャートを、Fig. 3-11 に示す。

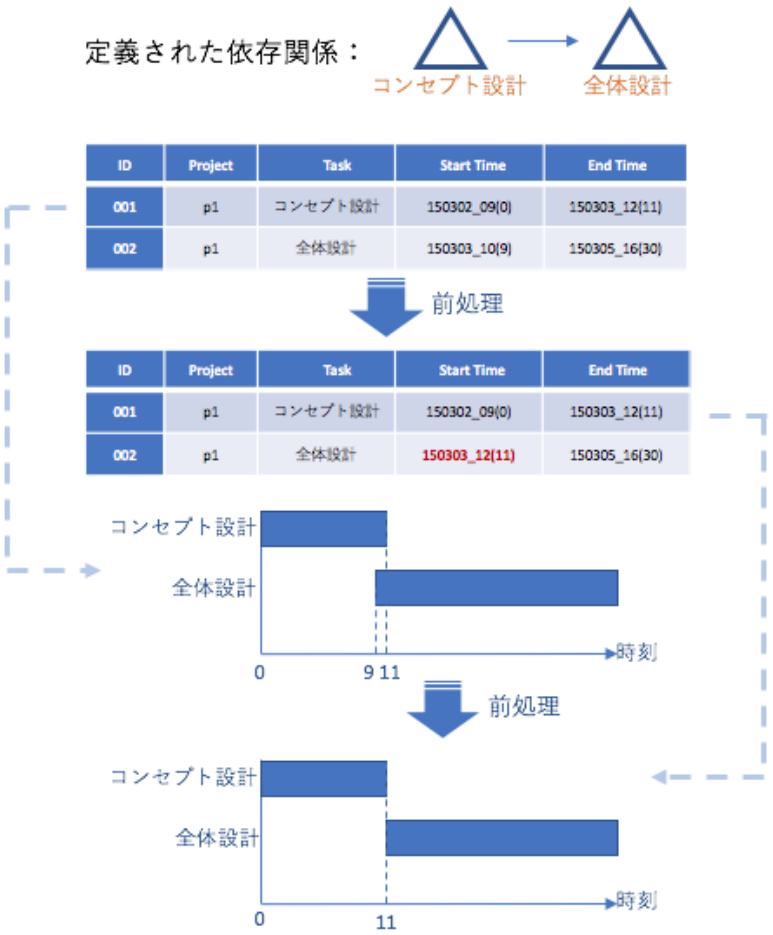

Fig. 3-11 前処理の例

この段階では、実績データを2つずつ比較し、依存関係に直接もしくは間接に反すれば、一定のルールに従い、処理を行う。また、1回の処理が実行され、つまり、データ全体において何らかの変化が生じれば、最初から前処理を行うこととする。

実績データ 2 件において、依存関係による前処理が必要となるシチュエーションが、Fig. 3-12 に示す 4 つのパターンがあると考えられる。ただし、上流タスクの開始時間が、下流タスクの開始時間より遅くなった場合、作業が重ねている部分にだけ、処理を行う(Fig. 3-12 のパターン 3 とパターン 4 を参照)。

Fig. 3-7 に示した実績データの例において、依存関係による前処理を行った後のデータを、Table 3-8 に示す。赤塗りした部分は、Table 3-7 から変更された部分を示す。

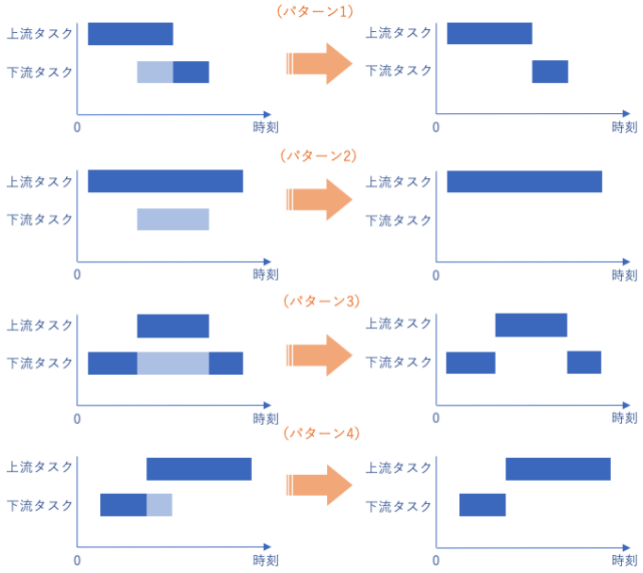

Fig. 3-12 依存関係による前処理のパータン

Table 3-8 依存関係による前処理を終えたデータ：プロジェクト p1

ID	Project	Task	Start Time	End Time	Resource Info	Ending Status
001	p1	コンセプト設計	0	11	5 工数/h	Normal
002	p1	全体設計	11	30	5 工数/h	Normal
003	p1	部品設計	32	55	5 工数/h	Normal
004	p1	原材料購入	55	57	10 工数/h	Rework:コンセプト設計
005	p1	部品購入	55	58	7 工数/h	Suspend
006	p1	全体設計	58	63	5 工数/h	Normal
007	p1	部品設計	63	66	7 工数/h	Normal
008	p1	原材料購入	66	71	7 工数/h	Normal
009	p1	部品購入	66	71	5 工数/h	Normal
010	p1	部品製造	71	101	20 工数/h	Rework:部品設計
011	p1	部品設計	103	106	5 工数/h	Normal
012	p1	原材料購入	106	111	2 工数/h	Normal
013	p1	部品購入	106	119	2 工数/h	Normal
014	p1	部品製造	119	131	20 工数/h	Normal
019	**p1**	**組み立て**	**131**	**146**	**8 工数/h**	**Suspend**
017	p1	部品製造	146	153	15 工数/h	Normal
015	p1	組み立て	153	160	8 工数/h	Normal
018	p1	塗装	161	172	3 工数/h	Normal

依存関係による前処理に関するメソッドの大まかな説明を Table 3-9 に示す。

Table 3-9 依存関係による前処理に関するメソッド

クラス	メソッド	戻り値の型	説明
Extraction	-preprocess_d()	void	全ての実績データに対して、依存関係による前処理を行う
Project	-preprocess_d()	Boolean	該当プロジェクトに関する実績データに対して、依存関係による前処理を行う。一度処理を行うと、true を返す；総回しが完了し、何の処理も行わなければ、false を返す。
	-getTaskByName(Data)	Task	全てのタスクの中から、実績データ Data が対応するタスクの情報を取得する
Task	-ifDependentTask(Task)	Boolean	Task が該当タスクの上流タスクである場合、true を返す；逆の場合、false を返す。

具体的に、Table 3-9 にある各メソッドのアルゴリズムは、以下の通りである:

- **Extraction.preprocess_d()**

```
For each Project p:
    While (true):
        If (p.preprocess() = false):
            break While;
        else:
            update Extraction.dataList by p.dataList;
        End If.
    End While.
```

End For.

- **Project.preprocess_d()**

```
For each Data d1 in Project.dataList:
    Task t1 = getTaskByName(d1);
    For each Data d2 in Project.dataList:
        Task t2 = getTaskByName(d2);
        if(t2.ifDependentTask(t1) = true and d1.st < d2.st):
            if (d2.st < d1.et and d2.et > d1.et):
                d2.st ← d1.et;
                Return true;
            End if.
            if (d2.st < d1.et and d2.et < d1.et):
                delete d2 from all data;
                Return true;
            End if.
        End if.
        if(t2.ifDependentTask(t1)=true and d1.st > d2.st):
            if (d2.et > d1.et):
                Data d3 ← d2;
                d3.st ← d2.st; d3.status ← "Suspend";
                add d3 to Project.dataList;
                d2.et ← d1.st;
                Return true;
            End if.
            if (d2.et > d1.st and d2.et < d1.et):
                d2.et ← d1.st;
                Return true;
            End if.
        End if.
    End For.
End For.
Return false;
```

- **Project.getTaskByName(Data d)**

```
For each Task t:
    if(t.name = d.task): Return t;
```

End For.

Return null;

- **Task.ifDependentTask()**

For each Task t in inputTaskList:

if (this = t2):

Return true;

else:

Return t2.ifDependentTask(t1);

Return false;

End For.

3.4.4.2 終了時ステータスによる前処理

3.4.4.1 の処理を行った後に、終了時ステータスによる前処理を行う。この処理では、各タスクの実行時における工数が計算され、実行回数もカウントされる。

3.4.4.1 と同様に、終了時ステータスによる前処理も、プロジェクト毎に対する処理である。終了時ステータスが「Suspend」のデータに対して、同じプロジェクトにおける次のデータと結合し、工数が加算される。また、次のデータの終了時ステータスも「Suspend」である場合、さらに次のデータと結合し、終了時ステータスが「Normal」もしくは「Rework」のデータになるまで、処理を続行する。

家具の設計と製造プロジェクトの例では、終了時ステータスによる前処理を経過したデータを、Table 3-10 に示す。

Table 3-10 終了時ステータスによる前処理を終えたデータ：プロジェクト p1

ID	Project	Task	Work Amount	Occurrence Time	Ending Status
001	p1	コンセプト設計	55 工数	1	Normal
002	p1	全体設計	95 工数	1	Normal
003	p1	部品設計	115 工数	1	Normal
004	p1	原材料購入	20 工数	1	Rework:コンセプト設計
006	p1	全体設計	25 工数	2	Normal
007	p1	部品設計	21 工数	2	Normal
008	p1	原材料購入	35 工数	2	Normal
009	p1	部品購入	46 工数	1	Normal
010	p1	部品製造	600 工数	1	Rework:部品設計
011	p1	部品設計	15 工数	3	Normal
012	p1	原材料購入	10 工数	3	Normal
013	p1	部品購入	26 工数	2	Normal
014	p1	部品製造	240 工数	2	Normal
017	p1	部品製造	105 工数	3	Normal
015	p1	組み立て	176 工数	1	Normal
018	p1	塗装	33 工数	1	Normal

終了時ステータスによる前処理に関するメソッドの大まかな説明を Table 3-11 に示す。

Table 3-11 終了時ステータスによる前処理に関するメソッド

クラス	メソッド	戻り値の型	説明
Extraction	-preprocess_s()	void	全ての実績データに対して、終了時ステータスによる前処理を行う
Project	-preprocess_s()	void	該当プロジェクトに関する実績データに対して、終了時ステータスによる前処理を行う。
Data	-calcWorkAmount()	void	実績データ一件に対して、工数を計算する。

具体的に、Table 3-11 の各メソッドのアルゴリズムは、以下の通りである:

- **Extraction.preprocess_s()**

```
For each Project p:
    p.preprocess_s();
End For.
update Extraction.dataList by p.dataList;
```

- **Project.preprocess_s()**

```
Sort Data in p.dataList by Data.st;
For each d in dataList:
    d.calcWorkAmount();
    tcount ← 0; //タスクを数える変数
    count[] ← 0;//実行回数を数えるための配列
    For each t in taskList:
        tcount ← tcount+1;
        if(d.taskName = t.name):
            if(d.status <> "Suspend"): count[tcount-1] ← count[tcount-1]+1;
            d.oc ← count[tcount-1];
            break For.
        End if.
    End For.
End For.
For each Data d1 in dataList when d1.status <> "Suspend": //データの結合を行う
    For each Data d2 in dataList when d2.taskName=d1.taskName:
        if (d2.status = "Suspend" and d2.oc = d1.oc-1):
            d1.wa ← d1.wa + d2.wa;
            remove d2 from dataList;
        End if.
    End For.
End For.
```

- **Data.calcWorkAmount()**

```
Data.wa = (Data.et - Data.st) * Data.rc;
```

3.4.5 遅延モデルにおけるパラメータの抽出

引き続き、Table 3-7 の実績データの例を用いて、遅延モデルのパラメータの抽出手順を説明する。

ここで、Table 3-12 は、最後の列を除き、Table 3-7 が示した実績データを前処理したデータ、およびもう 2 つの家具設計と製造プロジェクトの実績データを示す。ただし、追加したプロジェクト p2、p3 の実績データも、同じく前処理を終えたデータである。

まず、各タスクに対して、最小工数の集計を行う。ここで、終了時ステータスが手戻りであるデータは、集計されないものとし、Table 3-12 では黄色で塗りつぶしている。各タスクの実行回数毎に、最小工数を探す。最小の工数を記録した実績データを、赤で塗りつぶす。次に、各実績データにおいて、遅延工数を算出する。遅延工数は、データ内の工数と最小工数の差分であり、floor 関数を用いて、正整数を取る。例では、Table 3-12 の最後列に遅延工数を示す。

次に、遅延確率の計算を行う。Table 3-12 における遅延確率計算に関する列を Table 3-13 の左側に示す。Table 3-13 では、プロジェクトやステータスなどの情報を除き、各実績データを、タスク、実行回数、遅延工数の優先順で並び替えている。タスクの実行回数ごとに、式(3.38)に従い、記録のある遅延工数に対して、確率を計算する。また、計算の結果を、Table 3-13 の最後列に示す。

$$dp_{i,o,wa} = count_{i,o,wa} / count_{i,o} \tag{3.38}$$

ここで、$count_{i,o}$とは、タスクt_iのo回目の実行における、手戻りが発生していなかった実績データの数であり、$count_{i,o,wa}$とは、タスクt_iのo回目の実行における、遅延工数がwaの実績データの数である。

Table 3-12 遅延パラメータの抽出例(1)

ID	Project	Task	Work Amount	Occurrence Time	Ending Status	Delay Work Amount
001	p1	コンセプト設計	55 工数	1	Normal	25 工数
002	p1	全体設計	95 工数	1	Normal	0 工数
003	p1	部品設計	115 工数	1	Normal	0 工数
004	p1	原材料購入	20 工数	1	Rework:コンセプト設計	-
006	p1	全体設計	25 工数	2	Normal	0 工数
007	p1	部品設計	21 工数	2	Normal	0 工数
008	p1	原材料購入	35 工数	2	Normal	0 工数
009	p1	部品購入	46 工数	1	Normal	0 工数
010	p1	部品製造	600 工数	1	Rework:部品設計	-
011	p1	部品設計	15 工数	3	Normal	0 工数
012	p1	原材料購入	10 工数	3	Normal	0 工数
013	p1	部品購入	26 工数	2	Normal	0 工数
014	p1	部品製造	240 工数	2	Normal	0 工数
017	p1	部品製造	105 工数	3	Normal	0 工数
015	p1	組み立て	176 工数	1	Normal	0 工数
018	p1	塗装	33 工数	1	Normal	5 工数
020	p2	コンセプト設計	30 工数	1	Normal	0 工数
021	p2	全体設計	156 工数	1	Normal	61 工数
022	p2	部品設計	180 工数	1	Normal	65 工数
023	p2	原材料購入	10 工数	1	Normal	0 工数
024	p2	部品購入	75 工数	1	Normal	30 工数
025	p2	部品製造	950 工数	1	Normal	0 工数
026	p2	組み立て	216 工数	1	Normal	40 工数
027	p2	塗装	28 工数	1	Normal	0 工数
028	p3	コンセプト設計	33 工数	1	Normal	3 工数
029	p3	全体設計	115 工数	1	Normal	20 工数
030	p3	部品設計	140 工数	1	Normal	25 工数
031	p3	原材料購入	15 工数	1	Normal	5 工数
032	p3	部品購入	76 工数	1	Normal	30 工数
033	p3	部品製造	960 工数	1	Normal	10 工数
034	p3	組み立て	216 工数	1	Normal	40 工数
035	p3	塗装	35 工数	1	Normal	7 工数

Table 3-13 遅延パラメータの抽出例(2)

ID	Task	Occurrence Time	Delay Work Amount	$dp_{i,o,wa}$
020	コンセプト設計	1	0 工数	$dp_{1,1,0} = 0.333$
028	コンセプト設計	1	3 工数	$dp_{1,1,3} = 0.333$
001	コンセプト設計	1	25 工数	$dp_{1,1,25} = 0.333$
002	全体設計	1	0 工数	$dp_{2,1,0} = 0.333$
029	全体設計	1	20 工数	$dp_{2,1,20} = 0.333$
021	全体設計	1	61 工数	$dp_{2,1,61} = 0.333$
006	全体設計	2	0 工数	$dp_{2,2,0} = 1$
003	部品設計	1	0 工数	$dp_{3,1,0} = 0.333$
030	部品設計	1	25 工数	$dp_{3,1,25} = 0.333$
022	部品設計	1	65 工数	$dp_{3,1,65} = 0.333$
007	部品設計	2	0 工数	$dp_{3,2,0} = 1$
011	部品設計	3	0 工数	$dp_{3,3,0} = 1$
023	原材料購入	1	0 工数	$dp_{4,1,0} = 0.5$
031	原材料購入	1	5 工数	$dp_{4,1,5} = 0.5$
008	原材料購入	2	0 工数	$dp_{4,2,0} = 1$
012	原材料購入	3	0 工数	$dp_{4,3,0} = 1$
009	部品購入	1	0 工数	$dp_{5,1,0} = 0.333$
024	部品購入	1	30 工数	$dp_{5,1,30} = 0.667$
032	部品購入	1	30 工数	
013	部品購入	2	0 工数	$dp_{5,2,0} = 1$
025	部品製造	1	0 工数	$dp_{6,1,0} = 0.5$
033	部品製造	1	10 工数	$dp_{6,1,10} = 0.5$
014	部品製造	2	0 工数	$dp_{6,2,0} = 1$
017	部品製造	3	0 工数	$dp_{6,3,0} = 1$
015	組み立て	1	0 工数	$dp_{7,1,0} = 0.333$
026	組み立て	1	40 工数	$dp_{7,1,40} = 0.667$
034	組み立て	1	40 工数	
027	塗装	1	0 工数	$dp_{8,1,0} = 0.333$
018	塗装	1	5 工数	$dp_{8,1,5} = 0.333$
035	塗装	1	7 工数	$dp_{8,1,7} = 0.333$

Table 3-14 に Table 3-12 のデータにおける遅延モデルのパラメータの抽出結果をまとめた。ただし、遅延工数が 0 である遅延確率の情報に関する記述は省略する。

Table 3-14 遅延モデル抽出結果の例

タスク名	遅延モデルのパラメータ	
	最小の工数	遅延確率
コンセプト設計	$mwa_{1,1} = 30$	$dp_{1,1,3} = 0.333$ $dp_{1,1,25} = 0.333$
全体設計	$mwa_{2,1} = 95$ $mwa_{2,2} = 25$	$dp_{2,1,20} = 0.333$ $dp_{2,1,61} = 0.333$
部品設計	$mwa_{3,1} = 115$ $mwa_{3,2} = 21$ $mwa_{3,3} = 15$	$dp_{3,1,25} = 0.333$ $dp_{3,1,65} = 0.333$
原材料購入	$mwa_{4,1} = 10$ $mwa_{4,2} = 35$ $mwa_{4,3} = 10$	$dp_{4,1,5} = 0.5$
部品購入	$mwa_{5,1} = 46$ $mwa_{5,2} = 26$	$dp_{5,1,30} = 0.667$
部品製造	$mwa_{6,1} = 950$ $mwa_{6,2} = 240$ $mwa_{6,3} = 105$	$dp_{6,1,10} = 0.5$
組み立て	$mwa_{7,1} = 176$	$dp_{7,1,40} = 0.667$
塗装	$mwa_{8,1} = 28$	$dp_{8,1,5} = 0.333$ $dp_{8,1,7} = 0.333$

遅延モデルにおけるパラメータの抽出に関するメソッドを Table 3-15 に示す。

Table 3-15 遅延モデルのパラメータ抽出に関するメソッド

クラス	メソッド	戻り値の型	説明
Extraction	-getMwa(Task)	void	引数である Task に対して、最小の工数を付与する。
	-getDelayP(Task)	void	引数である Task に対して、遅延確率を付与する。
	-getTaskByName(Data)	Task	全てのタスクの中から、実績データ Data が対応するタスクの情報を取得する

具体的に、Table 3-15 の各メソッドのアルゴリズムは、以下の通りである:

- **Extraction.getMwa(Task t)**

```
For each Integer 0<i<100:
    ifInit[] ← false; //タスクの工数が集計されたかどうかを記録する配列
    For each d in extraction.dataList:
        Task t = Extraction.getTaskByName(Data);
        If(d.oc=i and ifInit[Extraction.taskList.indexOf(t)]=false and d.status="Normal"):
            Add (i, d.wa) to t.mwa;
            ifInit[Extraction.taskList.indexOf(t)] ← true;
        Else if(d.oc=i and d.status="Normal")
            If(t.mwa(i)>d.wa):
                Add (i, d.wa) to t.mwa;
            End If.
        End If.
    End For.
    For each t in Extraction.taskList:
        If(ifInit[Extraction.taskList.indexOf(t)]=false):
            Add (i, 0) to t.mwa;
        End If.
    End For.
End For.
```

- **Extraction.getDelayP(Task t)**

```
For each t in Extraction.taskList:
    awaMap<Integer, List<Integer>> ← null; //実行回数ごとに、遅延工数を集計するマップ
    For each d in Extraction.dataList:
        awa ← Ceil(d.wa-Extraction.getTaskByName(d).mwa);
        If(awaMap.containsKey(d.oc)):
            Add awa to awaMap.get(d.oc);
        Else:
            Add (d.oc, List[awa]) to awaMap;
        End If.
    End For.
    For each Integer i in awaMap.keySet:
        Reversely sort awaMap.get(i); //遅延工数の降順でソートする
        p ← 1;
        tmp ← awaMap.get(0);
        For each key j in awaMap://確率の累積を計算し、Task.delay に与える
```

```
            If(awaMap.get(j)<tmp):
                Add (i, p, tmp) to t.delay;
            End If.
            p ← p – 1/ awaMap.get(i).size;
            tmp ← awaMap.get(i).get(j);
        End For.
    End For.
        For each Integer i from t.delay.oc: //確率の累積から、それぞれ遅延工数の確率を計算する
            Reversly sort all lists in t.delay by t.delay.probability;//確率の累積の降順でソート
            Double tp=1;
            For each Double key in t.delay.probability:
                tp ← tp – key;
                key ← tp;
            End For.
        End For.
End For.
```

- **Extraction.getTaskByName(Data d)**

```
For each Task t:
    if(t.name = d.task): Return t;
End For.
Return null;
```

3.4.6 手戻りモデルにおけるパラメータの抽出

手戻りモデルのパラメータの抽出手順について説明する。ここでも Table 3-12 の実績データ全体を取り上げる。手戻りモデルでは、進捗率に関する手戻り確率が計算されるため、まず、各タスクの各実績データに対して、記録時における進捗率を計算する。

Table 3-16 手戻りパラメータの抽出例

ID	Task	Work Amount	Minimum Work Amount	Occurrence Time	Ending Status	Progress When End
004	原材料購入	20 工数	10	1	Rework: コンセプト設計	2
023	原材料購入	10 工数	10	1	Normal	1
031	原材料購入	15 工数	10	1	Normal	1.5
010	部品製造	600 工数	950	1	Rework: 部品設計	0.6
025	部品製造	950 工数	950	1	Normal	1
033	部品製造	960 工数	950	1	Normal	1

手戻りパラメータの計算に関わる実績データを、Table 3-16 に示す。式(3.8)により計算されたデータ記録時の進捗率を、最右列に示す。手戻り確率は、式(3.39)に従い、計算する。ここで、$count_{i_1,o,pro,i_2}$は、タスクt_{i_1}がo回目の実行中にタスクt_{i_2}へ手戻りする時の、進捗率がproである実績データの数であり、$count_{i_1,o,\geq Pro}$は、手戻り発生の有無に関わらず、タスクt_{i_1}のデータ記録時進捗率がproより大きいもしくは等しい実績データの数である。

ただし、特定のタスクの特定実行回に対して、手戻りが発生したデータのみが記録された場合、最小の工数が遅延モデルパラメータの計算で算出できない。そのため、進捗率の計算が不可能となる。このようなタスクに対しては、データ記録時の進捗率を-1 とし、集計方法については、特別な処理を行わない。

$$rp_{i_1,o,pro,i_2} = count_{i_1,o,pro,i_2} / count_{i_1,o,\geq Pro} \tag{3.39}$$

説明例における手戻りパラメータの計算結果を Table 3-17 に示す。

Table 3-17 手戻りモデル抽出結果の例

タスク名	手戻りモデルのパラメータ
コンセプト設計	-
全体設計	-
部品設計	-
原材料購入	$rp_{4,1,2,1} = 1$
部品購入	-
部品製造	$rp_{6,1,0.6,3} = 0.333$
組み立て	-
塗装	-

手戻りモデルにおけるパラメータの抽出に関するメソッドと計算途中に用いるデータ構造を Table 3-18 に示す。

Table 3-18 手戻りモデルのパラメータ抽出に関するメソッドとデータ構造

クラス	メソッド	戻り値の型	説明
Extraction	-getRework(Task)	void	引数である Task に対して、手戻り確率を付与する。
クラス	データ構造		
ReworkCalc	-(Integer) oc: 手戻り時の実行回数 -(Integer) progress: 手戻り時のタスク進捗率 -(String) reworkDes: 手戻り先のタスク名		

具体的に、Table 3-18 にあるメソッドのアルゴリズムは、以下の通りである:

- **Extraction.getRework(Task t)**

For each d in Extraction.dataList: //終了時の進捗率を計算する

If(Extraction.getTaskByName(d).mwa.get(d.oc)):

*d.progressWhenEnd ← Floor(d.wa/Extraction.getTaskByName(d).mwa.get(d.oc)*10)/10;*

Else:

d.progressWhenEnd ← -1;

```
    End If.
End For.
For each t in Extraction.taskList:
    List<ReworkCalc> reworkCalcList ← null;
    For each d in Extraction.dataList: //ReworkCalc を用いて、手戻りに関する情報だけを取り出す
        If(d.task= t.name):
            Add (d.oc, d.progressWhenEnd, d.reworkTask) to reworkCalcList;
        End If.
    End For.
    i ← 1;
    ifContinue ← true; //実行回数が存在しない場合、次のタスクの手戻り確率計算へ
    While(ifContinue=true):
        ifContinue ← false;
        For each reworkCalc r1 in reworkCalcList:
            If(r1.oc=i):
                Map<Double, Integer> overProgressCount ← null;
                For each -1<p<30, step=0.1: //進捗率について集計する
                    For each reworkCalc r2 in reworkCalcList;
                        If(r2.oc=i and r2.progress>=p):
                            Add (p, overProgressCount.get(p)+1) to overProgressCount;
                        End If.
                    End For.
                End For.
                For each reworkCalc r2 in reworkCalcList: //手戻り先について集計する
                    count ← 0;
                    For each reworkCalc r3 in reworkCalcList:
                        If(r3.oc=i and r3.reworkDes = r2.reworkDes
and r3.progress=r2.progress):
                            count ← count+1;
                        End If.
                    End For.
```

If(count > 0):

Add (i, r2.progress, count/overProgressCount.get(progress), Extraction.getTaskByName(r2.reworkDes)) to t.rework;

End If.

End For.

ifContinue ← true;

i ← i+1;

End if.

End For.

End While;

End For.

3.4.7 プログラムの入力と出力

パラメータの抽出は、単独のプログラムにより実行される。実績データは、Fig. 3-13 のような形で入力される。また、プログラムの出力の例は、Fig. 3-14 に示す。

Fig. 3-13 抽出プログラムの入力例

result.xml

result.xml 〉 <NodeElementList>

```
<?xml version="1.0"?>
<!--Extracted Result-->
<NodeElementList>
    <TaskNode>
        <Name>コンセプト設計</Name>
        <MinimumWorkAmount>1,30.0</MinimumWorkAmount>
        <Delay>1,0.333,3</Delay>
        <Delay>1,0.333,25</Delay>
    </TaskNode>
    <TaskNode>
        <Name>全体設計</Name>
        <MinimumWorkAmount>1,95.0</MinimumWorkAmount>
        <MinimumWorkAmount>2,25.0</MinimumWorkAmount>
        <Delay>1,0.333,20</Delay>
        <Delay>1,0.333,61</Delay>
    </TaskNode>
    <TaskNode>
        <Name>部品設計</Name>
        <MinimumWorkAmount>1,115.0</MinimumWorkAmount>
        <MinimumWorkAmount>2,25.0</MinimumWorkAmount>
        <MinimumWorkAmount>3,15.0</MinimumWorkAmount>
        <Delay>1,0.333,25</Delay>
        <Delay>1,0.333,65</Delay>
    </TaskNode>
    <TaskNode>
        <Name>原材料購入</Name>
        <MinimumWorkAmount>1,10.0</MinimumWorkAmount>
        <MinimumWorkAmount>2,35.0</MinimumWorkAmount>
        <MinimumWorkAmount>3,10.0</MinimumWorkAmount>
        <Delay>1,0.5,5</Delay>
        <Rework>1,2.0,1.0,コンセプト設計</Rework>
    </TaskNode>
```

Fig. 3-14 抽出プログラムの出力例

3.5 パラメータ不足となる状況での処理

抽出されたパラメータとシミュレーションモデルの不確実性パラメータは、同様な型を持つが、実績データの多様性を考えると、抽出情報が不足する可能性がある。本節では、不確実性情報が不足している状況における例外処理について、説明する。

シミュレーションを実行する時に、タスクの特定実行回において、最小工数の情報がない場合が生じ得る。この状況および発生する原因を、Fig. 3-15 に示す例で説明する。

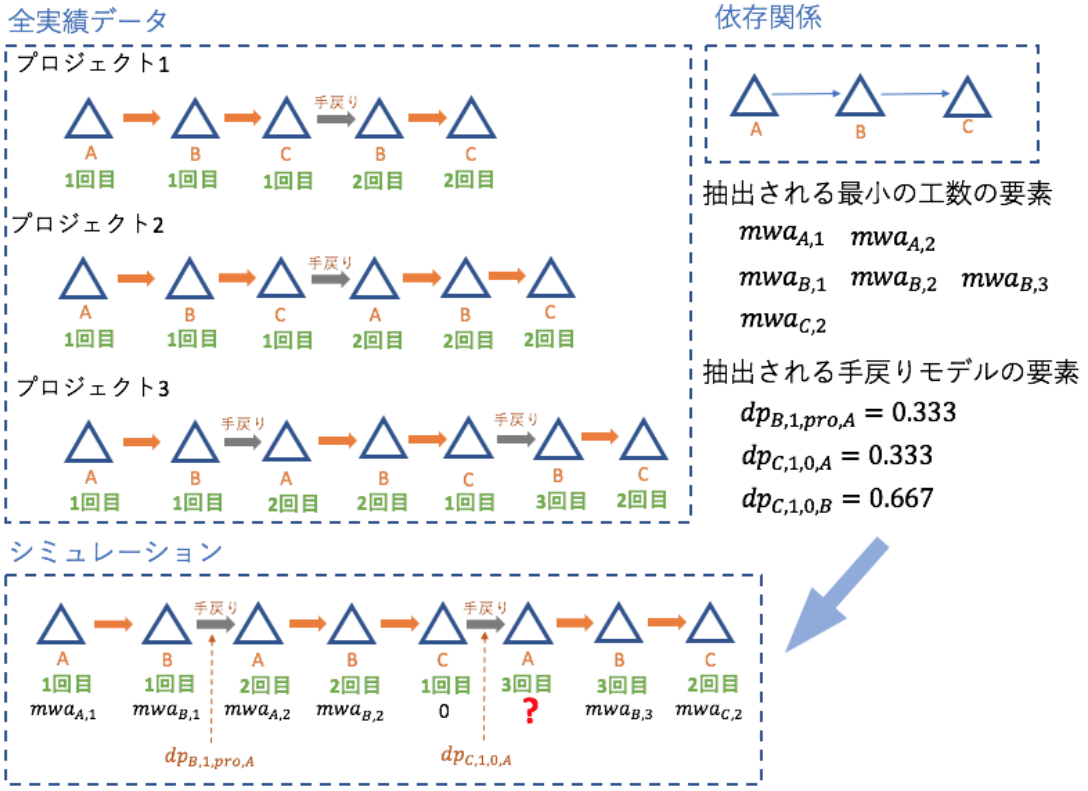

Fig. 3-15 例外処理が必要とされる状況

ここで、A、B、C の 3 つのタスクが、右上に示すような線形な依存関係を持つことを想定し、過去に、同じタスクとタスク間依存関係を持つプロジェクトが 3 つ実行されたことを仮定する。全ての実績データでの作業順番を、左上の「全実績データ」の枠に示す。全実績データから、右側に示した最小の工数と手戻りモデルの要素を、抽出することが可能である。ここで、シミュレーションの進行中に、タスク B の1回目実行中にタスク A への手戻り、およびタスク C の1回目実行中にタスク A への手戻りの両方とも発生となった場合、タスク A が3回目に実行される。しかし、タスク A は3回目実行されたことがなく、最小の工数に関する情報が不足となる。この状況では、提案手法によって、予想できなかった手戻りが発生したと認識し、特別な処理を行わない。ただし、プログラム上では、最小工数における情報が不足する場合、該当タスクの特定実行回数の工数を 0 とする。

3.6 開発したシミュレータ

3.6.1 概要

本研究では、提案するシミュレーションモデルを搭載したシミュレータを使用し、シミュレーションを行う。該当シミュレータは、満行ら[25]が開発した船舶建造シミュレータに基づき、機能の追加、削除を行うことで開発した。本節では、シミュレータのユーザインターフェイスについて紹介し、入力、実行および出力の各手順も、詳細に説明する。

3.6.2 シミュレータの入力と実行

シミュレータの主なユーザインターフェイスを Fig. 3-16 と Fig. 3-17 に示す。Fig. 3-16 と Fig. 3-17 は、それぞれタスクに関する情報を入力するためのインターフェイスとリソース、リソースチームに関する情報を入力するためのインターフェイスである。

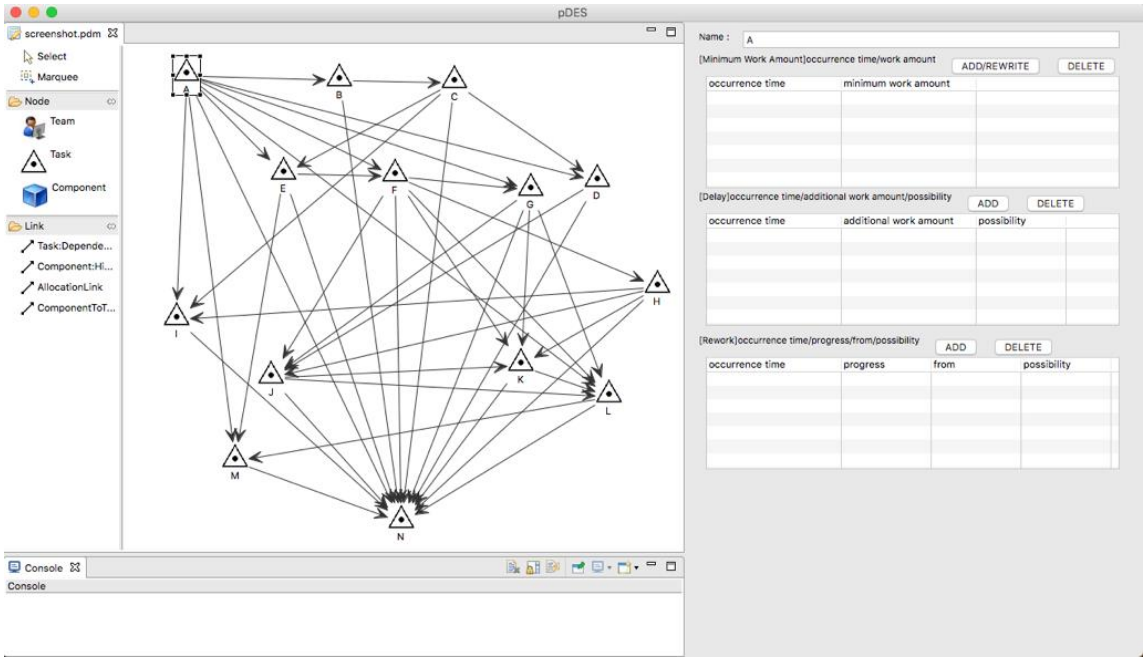

Fig. 3-16 タスク情報の入力インターフェイス

タスク情報の入力においては、タスク間依存関係、最小の工数、遅延モデルのパラメータ

および手戻りモデルのパラメータの 4 つのカテゴリーがある。左側の「Node」にある「Task」をクリックし、Task ノードを描画することができる。次に、左側にある「Task:Dependency」を選択し、Task ノード間の依存関係を示す矢印を描画することができる。ただし、矢印が指しているタスクは、下流タスクである。描画エリアで編集するタスクを選択すると、右側に最小の工数、遅延モデルのパラメータと手戻りモデルのパラメータを入力することができるインターフェースが表示される。

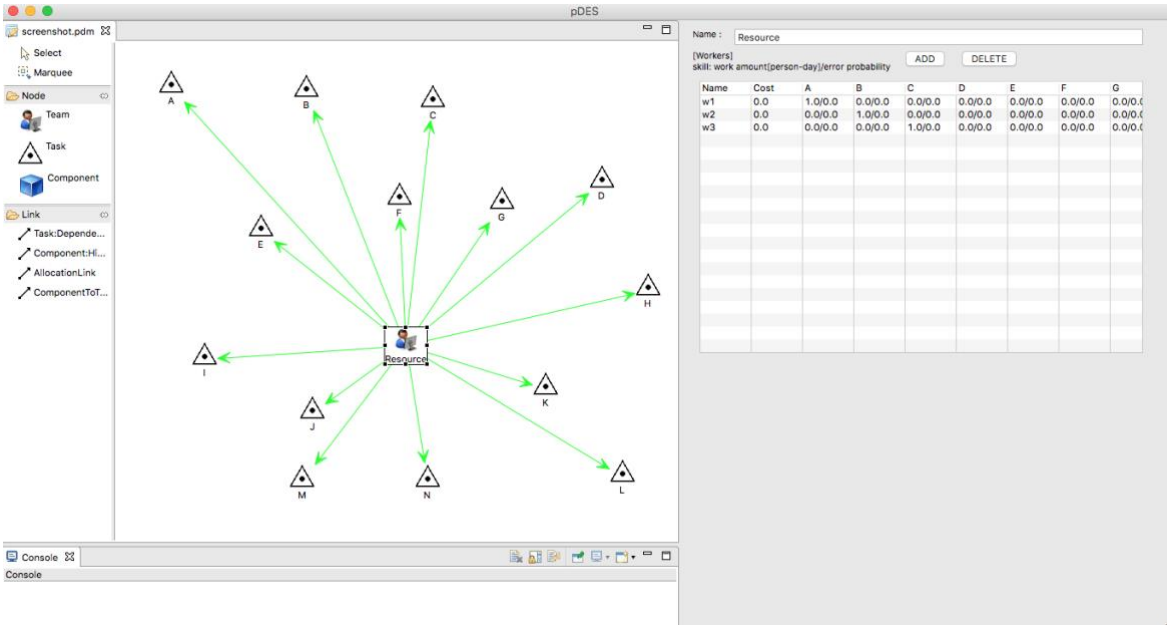

Fig. 3-17 リソース情報の入力インターフェイス

リソース情報の入力において、各リソースチームのスキル値とタイムステップ毎のコスト（省略することも可能）がある。左側の「Node」にある「Team」をクリックし、リソースを描画することができる。ただし、1 つの Team ノードは、提案手法における全てのリソースチームを含める。左側の「AllocationLink」を選択し、「Team」ノードから、全てのタスクに向けて、矢印を引く。また、「Team」ノードを選択すると、右側にて各リソースチームのスキル値とコストを入力することができる。

シミュレーションするプロジェクトの各ノードと矢印を入力した後に、シミュレータのメニューにある「File」の中で、「Open extracted parameter file」を選択することによって、3.4.7 で紹介したパラメータ抽出プログラムによる xml 形式の出力ファイルにある各不確実性パラメータを、そのまま導入することが可能である。

3.6.3 シミュレータの出力

シミュレータのメニューから、「Simulation」「Run multiple」「TSLACK+Uncertainty Model」の順に選択し、シミュレーション回数を入力すると、シミュレーションを実行することができる。出力として、各回シミュレーションにおけるタスクとリソースの実行開始時間と完了時間が、csv ファイルの形で出力される。また、複数回シミュレーションの見積もり工期工期および要するコストをまとめた csv ファイルも出力される。

第4章 ケーススタディ

4.1 はじめに

本章では、ケーススタディを用いて、提案手法のプログラム上における挙動の検証した後、シミュレーションモデルの妥当性（Validation）について検討する。その後、提案手法のデモンストレーションを行う。

ケーススタディ 1 では、仮想の小規模プロジェクトを用いて、プログラムの挙動の検証、および提案したシミュレーションモデルの妥当性検討を行う。ただし、仮想プロジェクトの実績データは、T. R. Browning ら[2]が提案したシミュレーションモデルを使用し、生成したものである。また既存モデルで生成したデータを用いた妥当性の検討方法は、R. G. Sargent[45]により紹介された方法である。

ケーススタディ 2 では、同じく T. R. Browning ら[2]による先行研究でケーススタディとして挙げられたプロジェクトを用いて、提案手法のデモンストレーションを行う。ただし、デモンストレーションのシナリオとして、リソースに関する 2 つのストラテジーを提示し、シミュレーションを行った後、出力結果を用いて、最良策を選択する。

4.2 ケーススタディ 1：検証

4.2.1 概要

ケーススタディ1の概要を Fig. 4-1 に示す。ケーススタディ1では、まず、仮想プロジェクトに対して設定を行い、少量の実績データを、既存モデルを用いて生成する。ここで、実績データを少量にする理由として、データの数を少なくすることによって、解析的に提案手法の諸計算を行えるようになるからである。次に、実績データから、プログラムによるパラメータ抽出結果と、解析的に計算した抽出結果を比較し、パラメータの抽出について検証する。シミュレーションモデルに対しても、同様に解析的に計算した出力結果とシミュレーション結果を比較し、シミュレーションモデルの正しい挙動を検証する。最後に、実績データにおけるリソース情報と、抽出したパラメータを用いて、シミュレーションを実行し、シミュレーション結果を実績データと比較し、モデルの妥当性について記述する。

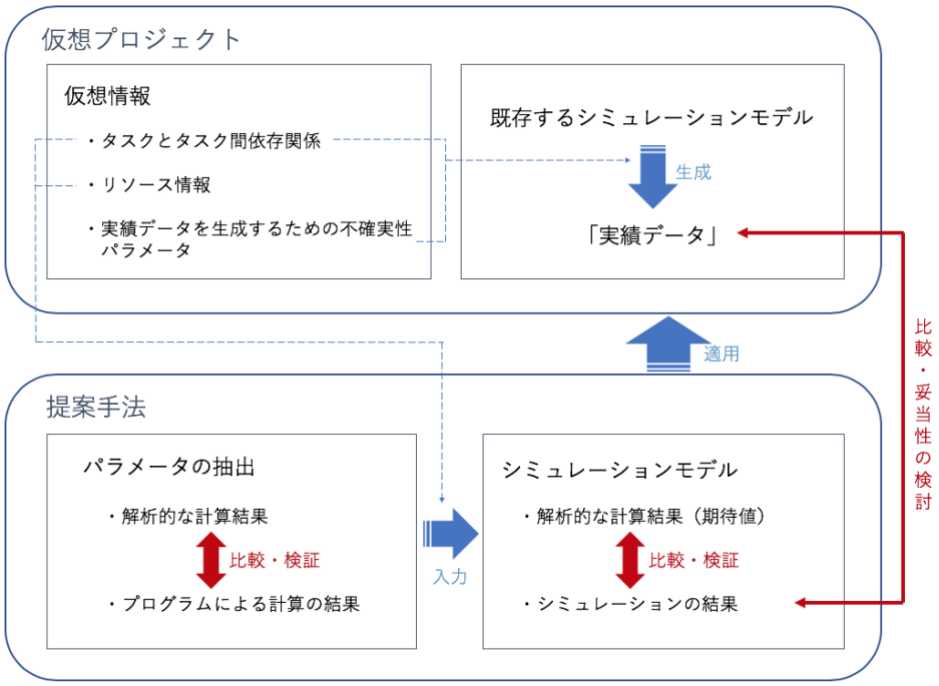

Fig. 4-1 ケーススタディ 1 の概要

4.2.2 仮想プロジェクトの設定

仮想プロジェクトは抽象的なプロジェクトであり、特定なシチュエーションを想定せず、少数のタスクとシンプルな依存関係のみを設定する。タスクとタスク間の依存関係、および検証するための設定の概要を、Fig. 4-2 に示す。

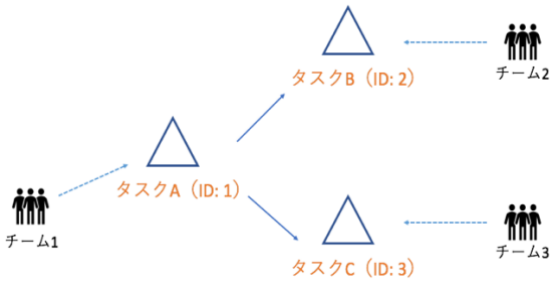

Fig. 4-2 リソース情報の入力

仮想プロジェクトは、3つのタスクから構成され、Table 4-1 に示す通りの依存関係を持つ。タスク A は、依存関係において、一番上流のタスクであるため、他タスクへの手戻りは発生し

ない。リソース情報は、Table 4-2 に示す通りである。実績データを生成するための既存モデルでは、工数ではなく、期間でシミュレーションするため、リソース情報は出力時のみに、一律「1 工数/日」と記述する。このような設定によって、提案手法では、実績データと同様なリソース情報を用いて、シミュレーションすることができる。

Table 4-1 仮想プロジェクトのタスク間依存関係

タスク名	ID	1	2	3
A	1		0	0
B	2	1		0
C	3	1	0	

Table 4-2 仮想プロジェクトにおけるリソースのスキル表

ID / スキル	1	2	3
A	1	0	0
B	0	1	0
C	0	0	1
			（工数/日）

4.2.3 生成した実績データ

4.2.1 で説明した通り、仮想プロジェクトの実績データは、先行研究における、遅延と手戻りの要素を含めたシミュレーションモデルから生成する。ここで、既存のモデルは、プロジェクトのリソースを考慮せず、各アクティビティー（タスク）の工期を用いて、プロジェクト全体の工期を見積もるためのモデルである。

仮想プロジェクトの実績データが 10 回分があるとし、既存モデルにより生成したデータを Fig. 4-3 のガントチャートに示す。各ガントチャートの左上にプロジェクトの番号であり、1 行目にある 1、2、 ... 、30 の数字は、それぞれ、各自のコマの左側枠線が代表する時刻である。コマに、タスク名が記述された箇所は、次の時刻から、記述されたタスクへ手戻りすることを表

し、「!」が記述された箇所は、作業の中断、つまり実績データ中の「Suspend」状態を表す。また、実績データを生成するための既存モデルに対する入力パラメータと既存モデルについての紹介は、Appendix A に参照する。

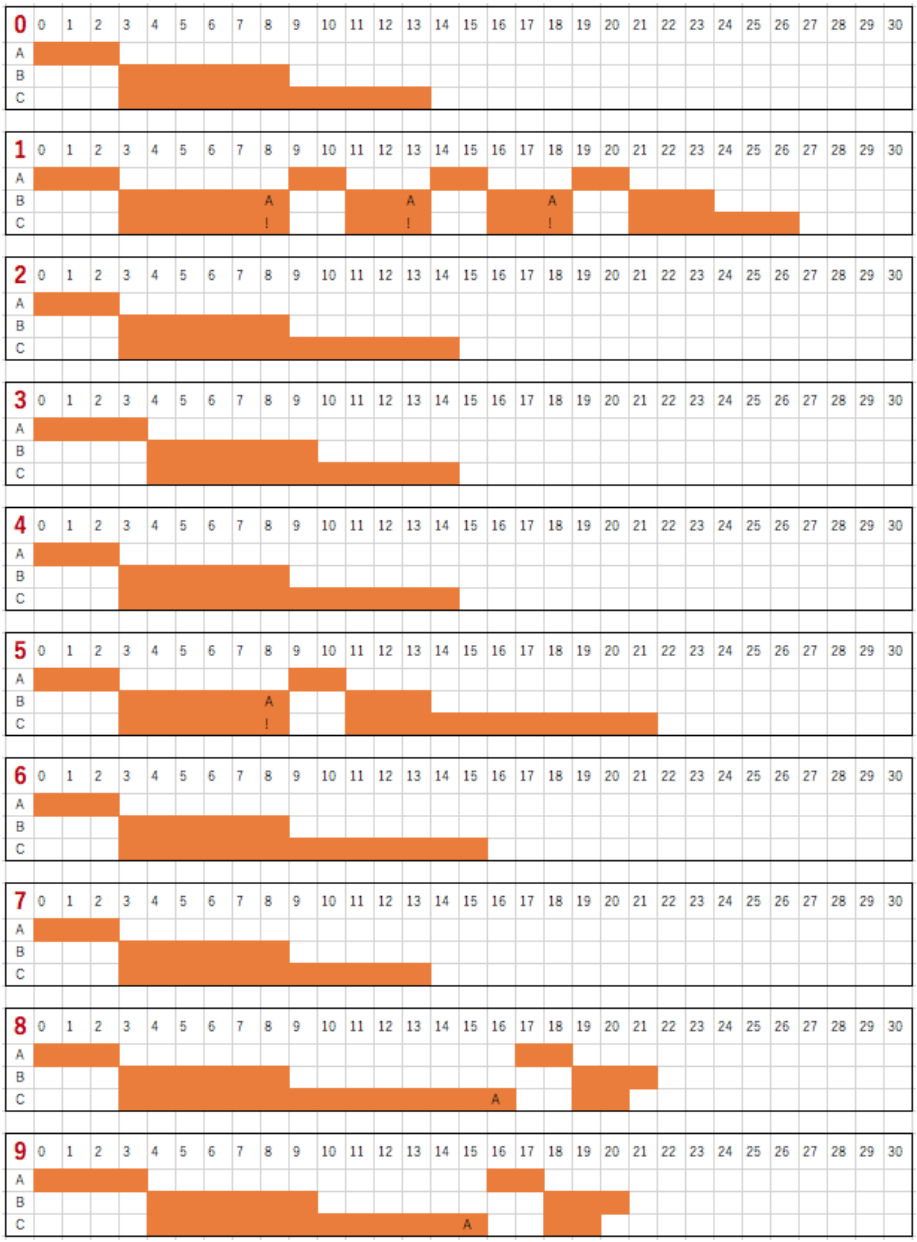

Fig. 4-3 生成した仮想プロジェクトの実績データのガントチャート

4.2.4 解析的計算とプログラムによるパラメータ抽出の結果

3.4 節の内容に従い、計算と集計によるパラメータの抽出結果を Table 4-3 に示す。また、プログラムにより抽出結果を Fig. 4-4 に示す。両者の情報が一致することにより、パラメータ抽出プログラムの正しい挙動が検証された。ただし、パラメータ抽出結果においては、遅延しないもしくは手戻りしない確率を省略する。

result.xml

result.xml 〉 No Selection

```
<?xml version="1.0"?>
<!--Extracted Result-->
<NodeElementList>
    <TaskNode>
        <Name>A</Name>
        <MinimumWorkAmount>1,3.0</MinimumWorkAmount>
        <MinimumWorkAmount>2,2.0</MinimumWorkAmount>
        <MinimumWorkAmount>3,2.0</MinimumWorkAmount>
        <MinimumWorkAmount>4,2.0</MinimumWorkAmount>
        <Delay>1,0.19999999999999996,1</Delay>
    </TaskNode>
    <TaskNode>
        <Name>B</Name>
        <MinimumWorkAmount>1,6.0</MinimumWorkAmount>
        <MinimumWorkAmount>2,3.0</MinimumWorkAmount>
        <MinimumWorkAmount>4,3.0</MinimumWorkAmount>
        <Rework>1,1.0,0.2,A</Rework>
        <Rework>2,1.0,0.25,A</Rework>
        <Rework>3,-1.0,1.0,A</Rework>
    </TaskNode>
    <TaskNode>
        <Name>C</Name>
        <MinimumWorkAmount>1,11.0</MinimumWorkAmount>
        <MinimumWorkAmount>2,2.0</MinimumWorkAmount>
        <Delay>1,0.09999999999999998,7</Delay>
        <Delay>1,0.09999999999999998,6</Delay>
        <Delay>1,0.09999999999999998,3</Delay>
        <Delay>1,0.09999999999999998,2</Delay>
        <Delay>1,0.29999999999999993,1</Delay>
        <Rework>1,1.2,0.3333333333333333,A</Rework>
        <Rework>1,1.0,0.1,A</Rework>
    </TaskNode>
</NodeElementList>
```

Fig. 4-4 プログラムによる仮想プロジェクトのパラメータ抽出結果

Table 4-3 計算による仮想プロジェクトのパラメータ抽出結果

タスク名	遅延モデルのパラメータ	手戻りモデルのパラメータ
A	$mwa_{A,1}=3$ $mwa_{A,2}=2$ $mwa_{A,3}=2$ $mwa_{A,4}=2$ $dp_{A,1,1}=0.2$	-
B	$mwa_{B,1}=6$ $mwa_{B,2}=3$ $mwa_{B,4}=3$	$rp_{B,1,1,A}=0.2$ $rp_{B,2,1,A}=0.25$ $rp_{B,3,-1,A}=1$
C	$mwa_{C,1}=11$ $mwa_{C,2}=2$ $dp_{C,1,1}=0.3$ $dp_{C,1,2}=0.1$ $dp_{C,1,3}=0.1$ $dp_{C,1,6}=0.1$ $dp_{C,1,7}=0.1$	$rp_{C,1,1,A}=0.1$ $rp_{C,1,1.2,A}=0.333$

4.2.5 解析的計算とプログラムによるシミュレーションの結果

各入力値に基づき、3.3 節と3.5 節の説明に従い、解析的に計算したシミュレーション結果と計算の過程を、Fig. 4-5 から Fig. 4-8 に示す。各説明図では、方形がシステムの状態を示し、青の方形が未完成の状態であり、赤の方形が完了状態である。また、各矢印は状態間の遷移ルートを示す。黄色い矢印は判定によるシステムの変化を示し、確率の値が付与される。グレーの矢印は、タイムステップの前進を表し、前進したタイムステップの値が付与される。なお、青の方形の横には、遅延と手戻りの発生以外に、システムの状態の詳細も記述される。システム状態の記述ルールを式(4.1)に示す。ここで、Fとは、タスクが完了と判定されたことを意味する。

$$タスク名\left(実行回数\right)=進行した工数/F \tag{4.1}$$

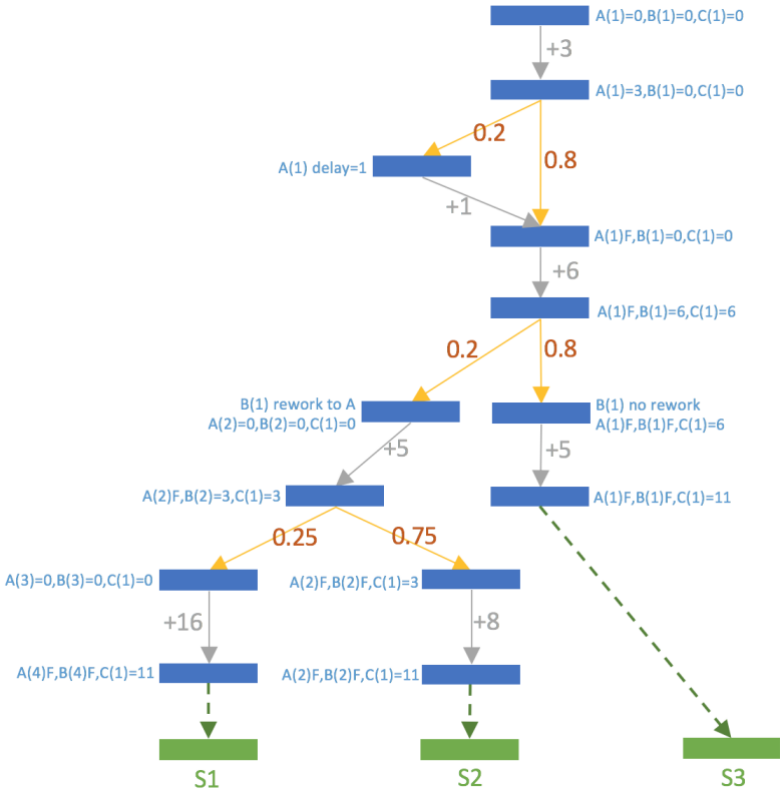

Fig. 4-5 シミュレーション結果の解析的計算過程 1

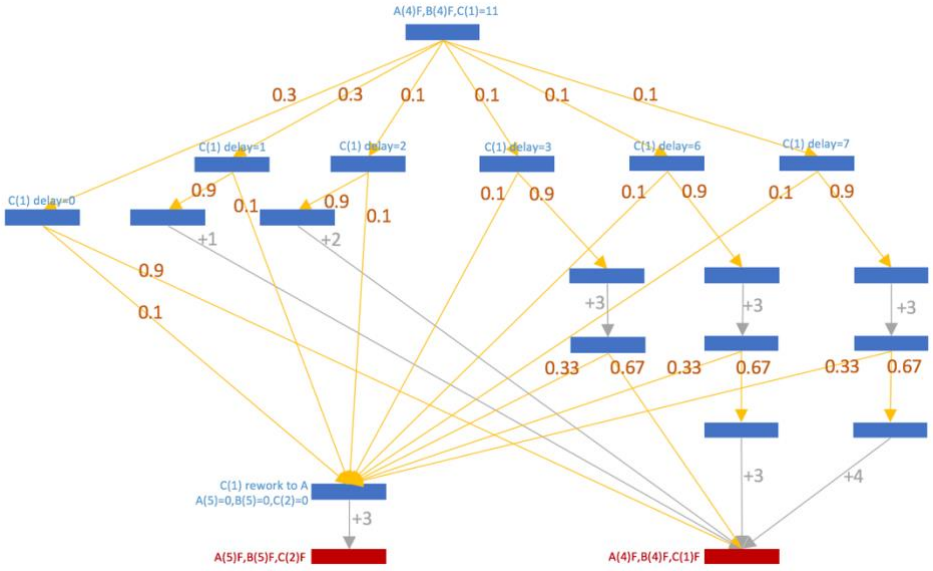

Fig. 4-6 シミュレーション結果の解析的計算過程(S1)

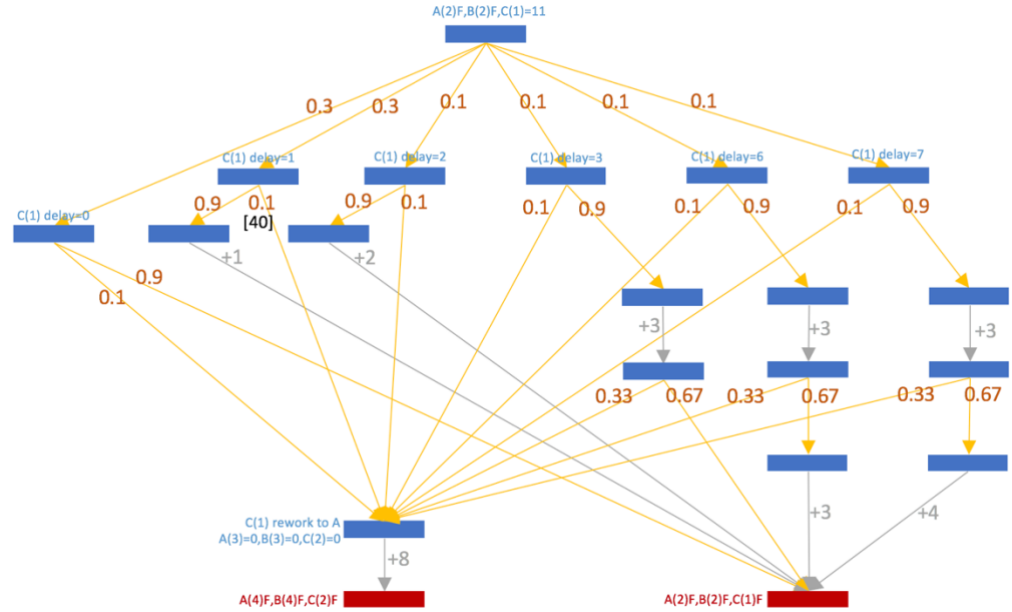

Fig. 4-7 シミュレーション結果の解析的計算過程(S2)

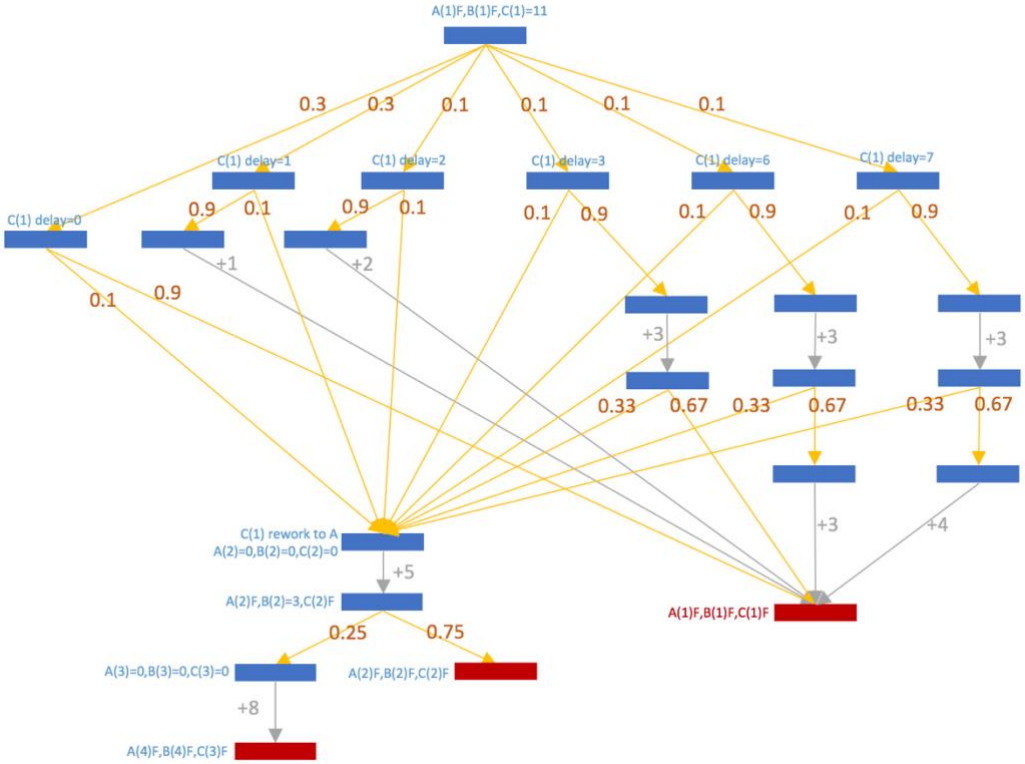

Fig. 4-8 シミュレーション結果の解析的計算過程(S3)

Fig. 4-5 では、タスク C の進捗率が1に到達する前の状態遷移を示し、Fig. 4-6、Fig. 4-7、Fig. 4-8 では、それぞれ、Fig. 4-5 と接続している状態 S1、S2、S3 から、可能なシステム状態遷移を示す。方形-矢印-方形の順に、全ての可能な状態遷移ルートを辿る。グレーの矢印の数値を累加した数値は理論上の見積もり工期であり、黄色の矢印の数値を累乗した数値は該当工期の結果が出力される確率である。

解析的計算による確率の計算結果とシミュレーション結果を Table 4-4 に示す。確率とシミュレーション結果における各工期の出現頻度と比較し、一致することによって、開発したシミュレータの正しい挙動が検証された。

Table 4-4 シミュレーション結果と確率

工期	5000 回シミュレーション内の頻度	確率	工期	5000 回シミュレーション内の頻度	確率
14	0.1674	0.1728	26	0.002	0.0018
15	0.2152	0.216	27	0.0136	0.016
16	0.1004	0.1008	28	0.0142	0.0112
17	0.0544	0.0528	29	0.0082	0.009
18	0.0094	0.0096	30	0.0386	0.039
19	0.0452	0.048	31	0.0212	0.0201
20	0.0546	0.0504	32	0.0072	0.0063
21	0.0502	0.048	33	0.016	0.0181
22	0.091	0.0852	34	0.0034	0.0043
23	0.0514	0.0513	36	0.004	0.006
24	0.0184	0.0189	37	0.0038	0.0039
25	0.009	0.0099	38	0.0012	0.0006

4.2.6 妥当性の検討

4.2.4 と4.2.5 では、それぞれ、抽出プログラムと開発したシミュレータの正しい挙動を検証した。4.2.6 では、実績データとシミュレーション結果の比較を用いて、提案したシミュレーショ

ンモデルの妥当性について検討する。同様なるリソース情報の前提で、実績データと提案手法によるシミュレーション結果の比較を、Table 4-5 に示す。

Table 4-5 仮想プロジェクトの実績データとシミュレーション結果の比較

工期	実績データ内の頻度	シミュレーション結果内の頻度	工期	実績データ内の頻度	シミュレーション結果内の頻度
14	0.2	0.1674	26	-	0.002
15	0.3	0.2152	27	0.1	0.0136
16	0.1	0.1004	28	-	0.0142
17	-	0.0544	29	-	0.0082
18	-	0.0094	30	-	0.0386
19	-	0.0452	31	-	0.0212
20	-	0.0546	32	-	0.0072
21	0.1	0.0502	33		0.016
22	0.2	0.091	34		0.0034
23	-	0.0514	36		0.004
24	-	0.0184	37		0.0038
25	-	0.009	38		0.0012

Table 4-5 から、シミュレーション結果では、実績データより、多くのバリエーションが示していることが判明した。その理由を以下に示す。

遅延確率と手戻り確率の組み合わせに、離散イベントシミュレーションによって、実績データより、多種多様な可能性を示した。例えば、「タスク A の 1 回目実行に遅延が発生」と、「タスク C の 1 回目の実行に遅延が発生しない」の組み合わせが、システムの状態遷移として可能である。つまり、実績データにない「タスク A の工数が 4」と「タスク C の工数が 11」という組み合わせを示すことができる。

よって結果のバリエーションは、シミュレーションモデルが算出された各パラメータに基づき、実世界で起きていなかったものの起こりうることを表現したと考えられる。また、提案手法が作業の遅延と手戻りについてを再現できたため、提案したシミュレーションモデルが妥当である。

4.3 ケーススタディ２：デモンストレーション

4.3.1 概要

ケーススタディ 2 は先行研究[2]に研究されたプロジェクトを用いたデモンストレーションケースであるため、まず、先行研究にあるプロジェクトの背景と生成した実績データについて説明する。次に、これら実績データを用いて、デモンストレーションを行い、シミュレーション結果について説明する。

デモンストレーションのシナリオとして、人員に関するリソーススストラテジーを 2 つ提示し、それぞれ、抽出した不確実性パラメータと共に、シミュレーションモデルに入力する。開発したシミュレータによりシミュレーション結果が出力され、これらの結果を分析することによって、より適切なストラテジーを選択する。

4.3.2 プロジェクトの背景と実績データ

先行研究[2]が着目していたプロジェクトは、アメリカのボーイング社による、無人航空機の統合システムの予備設計プロジェクト（以下：設計プロジェクト）である。先行研究[2]では、実プロジェクトにある重要の情報を隠すために、一部の情報を省略し、もしくは改竄を加えている。また、該当プロジェクトでは、システムの予備設計が、決められた標準プロセスに従うものである。標準プロセスの各手順を以下に示し、また、タスク間の依存関係を Table 4-6 に示す。

1. 航空機設計要求と目的ドキュメントの用意（Prepare vehicle preliminary design requirements and objectives）
2. 予備設計におけるコンフィギュレーションの作成（Create vehicle preliminary design configuration）
3. 表面モデルと内部塗装の準備（Prepare surfaced models and internal drawings）
4. 空力解析と評価（Perform aerodynamics analyses and evaluation）
5. 初期構造ジオメトリ（3D モデル）の作成（Create initial structural geometry）
6. FEM 法に向けて構造ジオメトリを作成（Prepare structural geometry and notes for FEM）
7. 構造設計（Develop structural design conditions）
8. 重量と慣性解析（Perform weights and intertias anallyses）

9. 安定性と制御分析および評価（Perform stability and control analyses and evaluation）

10. フリーボディーダイアグラムと応用負荷の開発（Develop freebody diagrams and applied loads）

11. 内部負荷分布の確定（Estabilish internal load distributions）

12. 構造強度、剛性、寿命の評価（Evaluate structural strength, stiffness and life）

13. 予備製造計画と分析（Preliminary manufacturing planning and analyses）

14. まとめと提案の準備（Prepare proposal of the developed vehicle）

Table 4-6 設計プロジェクトのタスク間依存関係[2]

タスクID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1		0	0	0	0	0	0	0	0	0	0	0	0	0
2	1		0	0	0	0	0	0	0	0	0	0	0	0
3	0	1		0	0	0	0	0	0	0	0	0	0	0
4	1	0	1		0	0	0	0	0	0	0	0	0	0
5	1	0	1	0		0	0	0	0	0	0	0	0	0
6	1	0	0	0	1		0	0	0	0	0	0	0	0
7	1	0	0	0	0	1		0	0	0	0	0	0	0
8	0	0	0	0	0	1	0		0	0	0	0	0	0
9	1	0	1	1	0	0	0	1		0	0	0	0	0
10	0	0	0	1	0	1	1	1	0		0	0	0	0
11	0	0	0	0	0	1	1	1	0	1		0	0	0
12	1	0	0	0	0	1	1	0	0	1	1		0	0
13	1	0	0	0	1	0	0	0	0	0	0	1		0
14	1	1	1	1	1	1	1	1	1	1	1	1	1	

本研究において、「複数件の実績データが存在する」ことが、提案手法を運用する前提であるため、該当プロジェクトと類似なシステムの開発設計プロジェクトが、10 件実行されたと仮定する。ケーススタディ 1 と同様に、先行研究[2]で提案されたシミュレーションモデルを 10 回実行し、シミュレーション結果を「実績データ」とする。実績データを生成するための既存モデルに対する入力パラメータは、Appendix B に参照する。

ここで、過去にある 10 件のプロジェクトにおけるリソース情報が同様であることを仮定し、実績データに、Table 4-7 に示したリソースの情報を与える（必要な機器や設備に関する情報は、検討する範囲に含まれないため、ここで省略する）。

Table 4-7 設計プロジェクトの実績データにおけるリソース情報[8]

タスク ID	リソース情報	スキル（工数/日）
1	統合製品開発チーム全体および機能スペシャリスト	15
2	構成デザイナー2 名	2
3	構成デザイナー1 名、構造デザイナー1 名	2
4	空力エンジニア 1 名	1
5	構造エンジニア 6 名、ストレスエンジニア 6 名	12
6	構造エンジニア 1 名、ストレスエンジニア 1 名	2
7	構造アナリスト 4 名	4
8	ウェイトエンジニア 2 名	2
9	制御エンジニア 1 名、構成エンジニア 1 名	2
10	空力エンジニア 1 名, FEM エンジニア 1 名, 一般負荷エンジニア 1 名	3
11	FEM エンジニア 2 名	2
12	強度エンジニア 4 名	4
13	製造エンジニア 4 名、調達担当者 1 名、製造エキスパート 8 名	13
14	統合製品開発チーム全体	10

生成された実績データのガントチャートを Fig. 4-9 に示す。これらの実績データを元にして、不確実性パラメータを抽出し、シミュレーションを用いて、リソースについて検討する。

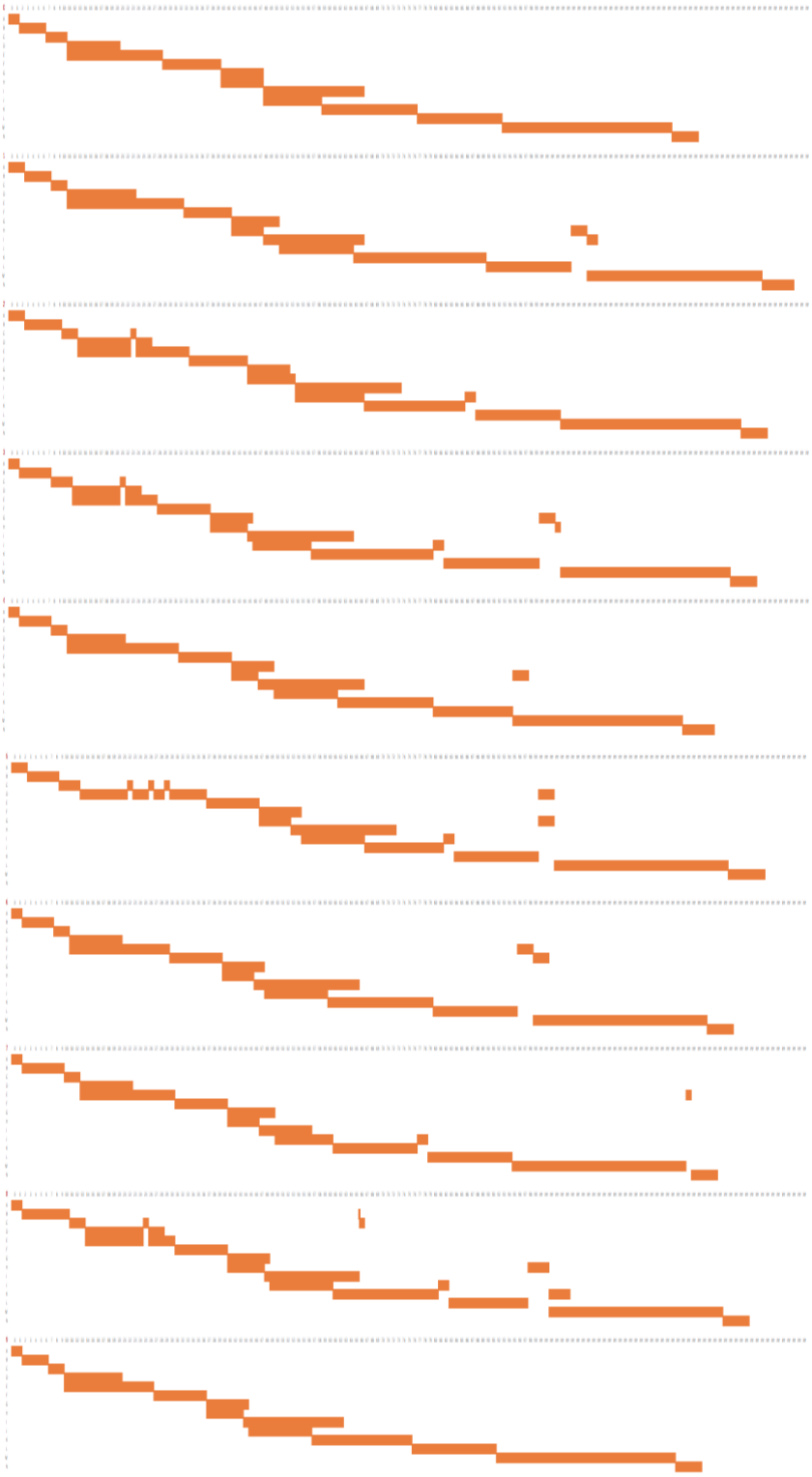

Fig. 4-9 生成した設計プロジェクトの実績データのガントチャート

4.3.3 検討するリソースストラテジー

提案手法のリソースに対する抽象化によって、リソースチーム内部の分割が難しいと想定したため、デモンストレーションでは、次のようなシチュエーションを想定する:今後のプロジェクトでは、各タスクを業務委託契約などにより、外部の組織に実行させる。また、複数の業務を行える外部組織もあれば、専門的に単一の業務しか行えない外部組織もある。

上記のシチュエーションを踏まえ、検討するリソースストラテジーは以下に示す:

- ストラテジー1: 単一のスキルを持つリソースチームをできるだけ多く採用する。
- ストラテジー2: 複数のスキルを持つリソースチームをできるだけ少なく採用する。

ストラレジーを比較するために、人員数について一定な制約を加える。プロジェクトに関わりうる人員を役割で分類し、種別毎に、人員種別の ID、および各種別において動員できる人数を Table 4-8 に羅列する。

Table 4-8 プロジェクトの人員と種別

種別 ID	役割	人員数
1	構成デザイナー	3
2	構造デザイナー	1
3	構造アナリスト	2
4	製造エキスパート	9
5	調達担当者	2
6	構成エンジニア	2
7	構造エンジニア	7
8	空力エンジニア	3
9	ストレスエンジニア	7
10	ウェイトエンジニア	2
11	制御エンジニア	2
12	FEM エンジニア	3
13	一般負荷エンジニア	2
14	強度エンジニア	2
15	製造エンジニア	5

Table 4-9 シミュレーションにおけるリソースチーム

ストラテジー1 におけるリソースチーム															
種別 ID / リソースチーム ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	6	0	6	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	8	1	0	0	0	0	0	0	0	0	0	4
ストラテジー2 におけるリソースチーム															
種別 ID / リソースチーム ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	0	0	0	0	0	0	6	0	6	0	0	0	0	0	0
12	0	0	0	8	1	0	0	0	0	0	0	0	0	0	4
13	2	1	0	0	0	0	0	1	0	0	0	0	0	0	0
14	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0
15	0	0	0	0	0	1	0	1	0	0	1	2	1	0	0

Table 4-10 設計プロジェクト：リソースストラテジー1

リソース ID / スキル（タスク ID）	1	2	3	4	5	6	7	8	9	10	11	12
1	15	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	2	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	12	0	0	0	0	0	0	0
6	0	0	0	0	12	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	2	0	0	0	0
10	0	0	0	0	0	0	0	0	3	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	1	0
13	0	0	0	0	0	0	0	0	0	0	0	13
14	15	0	0	0	0	0	0	0	0	0	0	0

Table 4-11 設計プロジェクト：リソースストラテジー2

リソース ID / スキル（タスク ID）	1	5	12	13	14	15
1	15	0	0	0	0	0
2	0	0	0	1	0	0
3	0	0	0	2	0	0
4	0	0	0	1	0	0
5	0	12	0	0	0	0
6	0	12	0	0	0	0
7	0	0	0	0	1	0
8	0	0	0	0	1	0
9	0	0	0	0	0	2
10	0	0	0	0	0	4
11	0	0	0	0	0	2
12	0	0	0	0	1	0
13	0	0	13	0	0	0
14	15	0	0	0	0	0

2 つのリソースストラテジーにおけるチーム構成と各リソースチームの ID を Table 4-9 に示す。ここで、チーム 1 から 12 は、ストラテジー1で動員するチームであり、チーム 1、チーム 5

およびチーム 12 から 15 は、ストラテジー2 で動員するチームである。各タスクに対するスキルは、実行するチーム内において、該当タスクを実行できる人数の値であり、「工数/日」の単位を付与する。ただし、実績データ中に明らかに多数の人員を要するタスクに対しては、ストラテジー構わず、同様な構成を持つチーム（チーム 1、チーム 5、チーム 12）を使用する。また、ストラテジー1 とストラテジー2 におけるリソースチームの各タスクに対するスキルを、Table 4-10 と Table 4-11 に示す。

各リソースチームのコストは、式(4.2)のように算出する。ここで、$cost_{rs_i}$とは、リソースチームrs_iの日当たりのコストであり、rsa_iは、rs_iが実行できるタスクの集合であり、式(4.3)のように表す。リソースチームのコストは、固定された人員費用のスキル値の加重平均との乗積である。

$$cost_{rs_i}(¥/day) = 10000 \times \frac{\sum_{k=0}^{N} s_{ik}}{|rsa_i|} \tag{4.2}$$

$$rsa_i = \{rs_i | s_{in} > 0, i = 1,2, \dots, N\} \tag{4.3}$$

4.3.4 シミュレーション結果

ストラテジー1 とストラテジー2 のリソース情報を入力し、それぞれ 1000 回シミュレーションを実行した結果を Fig. 4-10 と Table 4-12 に示す。また Fig. 4-10 では、1個の点につき、1 つのシミュレーション結果を表す。

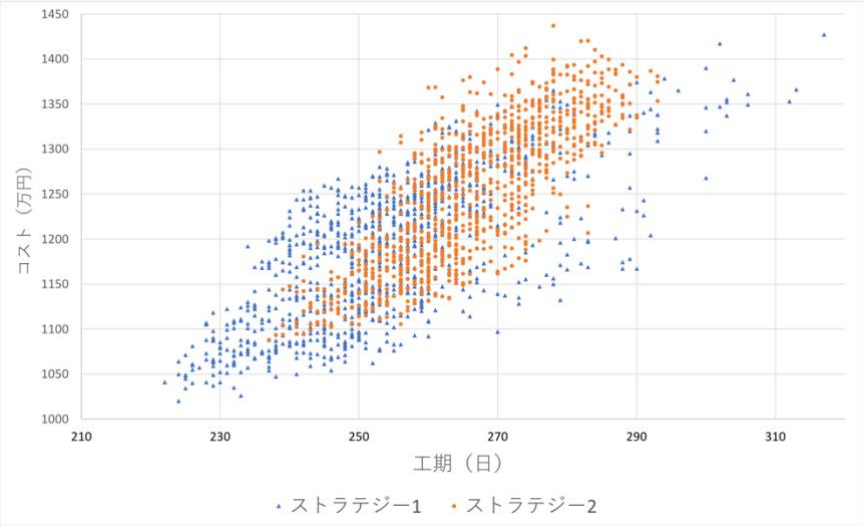

Fig. 4-10 設計プロジェクトにおける 1000 回シミュレーション結果

Table 4-12 設計プロジェクトにおける 1000 回シミュレーション結果

		ストラテジー1	ストラテジー2
工期	平均	255 日	266 日
	標準偏差	15.40 日	11.02 日
人員	平均	1188 万円	1252 万円
	標準偏差	80.00 万円	77.96 万円

全体的に、ストラテジー2 と比べて、ストラテジー1 では、リソースチームが分散されており、同時進行のできる作業に対して、リソースの競合が生じないため、平均的には、比較的短い見積もり工期と低いコストが出力された。一方ストラテジー2 では、元々リソース間の競合が激しく、並行作業がほぼできない状態であるため、手戻り作業の見積もり工期に対する影響が比較的小さい。このことから、ストラテジー2 における工期とコストの標準偏差はストラテジー1 より低くなるという結果が得られた。デモンストレーション結果からは、より短い納期と低いコストを目指すためには、ストラテジー1 を採用すべきであるということが分かる。

ここで、シミュレーション結果の中から、上記のストラテジー間の違いを示した結果の例を1つ取り上げ、Fig. 4-11 に示す。ここで、上部のガントチャートはストラテジー1 の結果例であり、下部のガントチャートはストラテジー2 の結果例である。ストラテジー1では、ID=7 のタスクと ID=8 のタスクはチーム 6 とチーム7により並行作業が可能であるが、ストラテジー2 では、チーム 14 のみがこの2つのタスクを実行できるため、タスクが先後に実行される。従って、工期が長引くことが分かった。

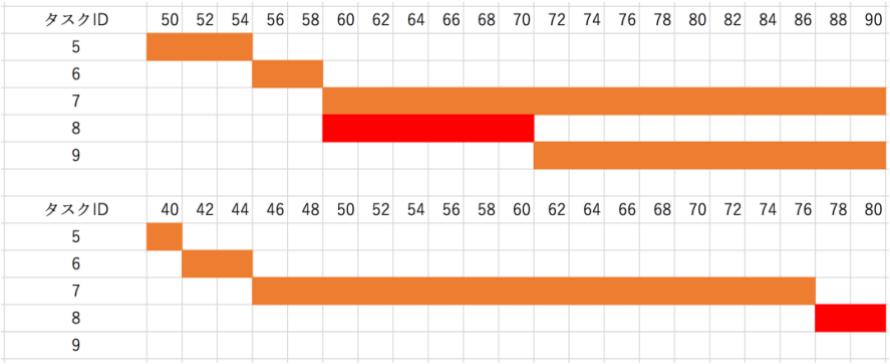

Fig. 4-11 ストラテジー間の違いを示した結果例

第5章 考察

5.1 はじめに

本章では、本研究の提案手法に対する考察を述べる。本研究で必要とされる実績データに関する考察と、提案手法の運用スコープに関する考察について述べる。

5.2 実績データに関する考察

本研究では、パラメータ抽出手法に適用する実績データの情報について定義した。プロジェクト名、タスク名、開始時刻、終了時刻および実行するリソースの情報は、一般的に、様々なプロジェクトで入手できる情報と考えられるが、タスク終了時の状態と手戻り発生時の手戻り先の情報は、必ずしも記録される情報ではない。そのため、本研究の進行にあたり、以下の試みをした。

・タスク終了時の状態と手戻り先情報を、入手できない情報とする。この場合、下流タスクの終了後すぐに、上流タスクがもう一度実行されることから、手戻りの発生として扱う。タスク数が少なく、依存関係もシンプルなプロジェクトにおいては、このような判断はできるが、プロジェクトの複雑度が高くなるにつき、手戻り情報が正確に計算される可能性が低くなり、適切性に欠ける手法となると考えられる。

・手戻り情報は実績データに含まれているが、依存関係やリソースの不足などによるタスクの中断に関する情報がない。この場合、パラメータの抽出は通常通りに行えるが、タスクの中断が考慮されていないため、特定実行回の最小工数が実際より著しく短くなる場合があり、シミュレーション上では、実世界のシチュエーションと顕著に離れている挙動が見られることがある。

以上の取り組みと考察に従って、本研究では、正確なパラメータを抽出するために、タスク終了時の情報や手戻り先の情報が、必要であるという結論に至った。

5.3 提案手法に関する考察

本研究で提案された、パラメータ抽出からシミュレーションモデルまでの一連の仕組みが適用できる状況を以下にまとめる。

・同じタスクとタスク間依存関係を持つプロジェクトが、過去に複数件実行された。

・プロジェクトにおいて、タスクとタスク間依存関係が、明確に DSM、もしくは類似する形

で、明確に定義された。

この 2 点を踏まえて、本研究の提案手法が適用できるプロジェクトが、標準プロセスを持つ、規模のばらつきのないプロジェクトと考えられる。また、特別受注など、一度のみ実行される大規模なプロジェクトにおいては、大量の作業のうちに、1 部だけが、標準プロセスに従うものであれば、適用できる可能性もある。

5.4 シミュレーションモデルの挙動の詳細について

ケーススタディ 1 では、シミュレーションモデルの挙動を確認したが、リソースの詳細の挙動については、解析的な結果を用いた検証はされていない。ここで、Fig. 5-1 と Fig. 5-2 を用いて、シミュレーションの出力結果では、リソースの挙動についても、詳しく確認できることを示す。

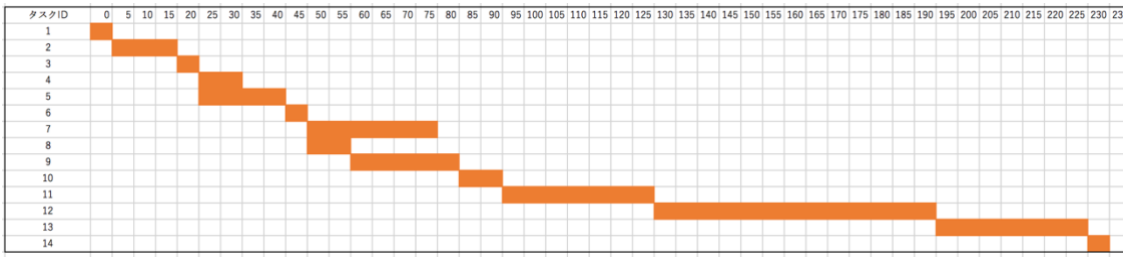

Fig. 5-1 リソース挙動の説明例（タスク）

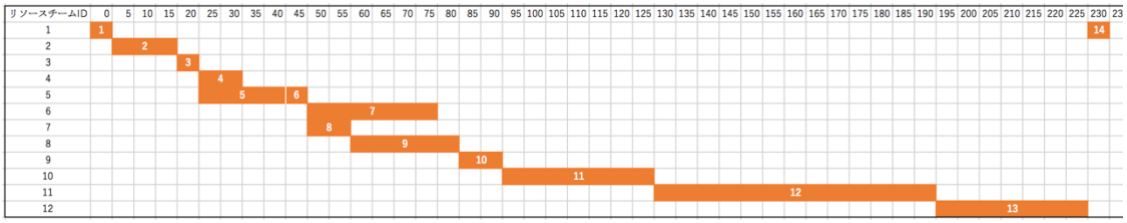

Fig. 5-2 リソース挙動の説明例（リソース）

Fig. 5-1 と Fig. 5-2 では、1個のシミュレーション結果を示す。Fig. 5-1 のガントチャートは、各タスクのガントチャートを示し、Fig. 5-2 のガントチャートは、各リソースのガントチャートを示す。またリソースのガントチャートでは、白の数字は、リソースが実行しているタスクの ID である。ただし、この結果は、ストラテジー1 のリソース情報による、手戻りが発生しないシミュレー

ションモデル結果である。このような出力結果を用いて、シミュレーションモデルのリソース挙動を確認することができた。

第6章 結論

6.1 結論

本研究では、実績データに基づき、手戻りと遅延を考慮したプロジェクト工期の見積もりを行う手法を提案した。提案手法では、実績データからのパラメータ抽出を想定した遅延と手戻りのモデルを提案した。実績データから、遅延工数とその確率、および各手戻りパターンの確率を算出し、得られたパラメータとリソース、タスクの依存関係をもとにシミュレーションを行うことで、工期を見積もる。

ケーススタディによって、提案手法の検証とデモンストレーションを行なった。ケーススタディ1では、提案手法に基づいたプログラムの正しい挙動を検証した上で、モデルの妥当性について検討した。ケーススタディ2では、既存のプロジェクトに対して、具体的なリソースに関するストラテジーを複数想定して提案手法を適用し、デモンストレーションを行った。デモンストレーションの結果、提案手法によってリソース情報の検討に関する意思決定を支援することが可能であること示した。

6.2 今後の展望

本研究の提案手法の適用には、同様なワークフローを持つプロジェクトの情報が、複数件存在することが、大きな前提である。また、ワークフロー、つまりタスクとタスク間依存関係が、明確に定義されなければならない。よって、考察で述べた内容も踏まえて、本研究における今後の方向性としては、以下の点があると考えられる。

・ワークフローが不明確なプロジェクトに対して、実績データから、タスクとタスク間依存関係を確定する手法の提案。

・全く同じワークフローを持つプロジェクトだけでなく、一部類似するプロジェクトにおいても、不確実性パラメータを抽出する手法の提案。

参考文献

[1]. C. Earl, J. Johnson, C. Eckert., 'Complexity' in *Design Process Improvement – A review of current practice* (2005): 174-197.

[2]. T. R. Browning, "Modelling and analyzing cost schedule and performance in complex system product development." Ph.D. Thesis, Massachusetts Institute of Technology (1998).

[3]. M. T. Pich, C. H. Loch, A. Meyer, "On uncertainty, ambiguity, and complexity in project management." *Management Science*, vol. 48-8 (2002): 1008-1023.

[4]. S. Globerson, "Impact of various work-breakdown structures on project conceptualization." *International Journal of Project Management*, vol. 12-3 (1994): 165-171.

[5]. Y. Akao, B. King, G. H. Mazur, "Quality function deployment: Integrating customer requirements into product design." *Productivity Press*, vol. 21 (1990).

[6]. D. V. Steward, "The design structure matrix: A method for managing the design of complex systems." *IEEE Transactions on Engineering Management*, vol. EM-28 (1981): 71-74.

[7]. T. R. Browning, "Applying the design structure matrix to system decomposition and integration problems: A review and new directions." *IEEE Transactions on Engineering management*, vol. 48-3 (2001): 292-306.

[8]. J. E. Kelly, "Critical-path planning and scheduling: Mathematical basis." *Operations Research*, vol. 9-3 (1961): 296-320.

[9]. W. Fazar, "Program evaluation and review technique." *The American Statistician*, vol. 13-2 (1959): 10.

[10]. A. A. B. Pritsker, "GERT: Graphical evaluation and review technique." Rand Corporation (1966).

[11]. Y. Jin, R. E. Levitt, "The virtual design team: A computational model of project organizations." *Computational & Mathematical Organization*

Theory, vol. 2-3 (1996): 171-195.

[12]. R. P. Smith, S. D. Eppinger, "A predictive model of sequential iteration in engineering design.", *Management Science*, vol. 43-8 (1997): 1104-1120.

[13]. T. R. Browning, S. D. Eppinger, "Modeling impacts of process architecture on cost and schedule risk in product development." *IEEE Transactions on Engineering Management*, vol. 49-4 (2002): 428-442.

[14]. R. Kolisch, S. Hartmann, "Heuristic algorithms for the resource-Constrained project scheduling problem: Classification and computational analysis." in *Project Scheduling* (1999): 147-178.

[15]. P. Brucker, A. Drexl, R. Mohring, K. Neumann, E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods." *European Journal of Perational Research*, vol. 112-1 (1999): 3-41.

[16]. M. Montazeri, L. N. Van Wassenhove, "Analysis of scheduling rules for an FMS." *The International Journal of Production Research*, vol. 28-4 (1990): 785-802.

[17]. B. Grabot, L. Geneste, "Dispatching rules in scheduling dispatching rules in scheduling: A fuzzy approach." *The International Journal of Production Research*, vol. 32-4 (1994): 903-915.

[18]. H. L. Gantt, "A graphical daily balance in manufacture." *Transactions of the American Society of Mechanical Engineers*, vol. XXIV (1903): 1322-1336.

[19]. D. C. Wynn, C. M. Eckert, "Perspectives on iteration in design and development." *Research in Engineering Design*, vol. 28-2 (2017): 153-184.

[20]. S. Cho, S. D. Eppinger, "A simulation-based process model for managing complex design projects." *IEEE Transactions on Engineering Management*, vol. 52-3 (2005): 316-328.

[21]. M. Jahangirian, T. Eldabi, A. Naseer, L. K. Stergioulas, T. Young, "Simulation in manufacturing and business: A review." European Journal of Operational Research, vol. 203 (2010): 1-13.

[22]. B. R. Moser, "The design of global work: Simulation of performance

including unexpected impacts of coordination across project architecture." Ph. D. Thesis, The University of Tokyo (2012).

[23]. S. Salimi, M. Mawlana, A. Hammad, "Performance analysis of simulation-based optimization of construction projects using high performance computing." *Automation in Construction*, vol. 87 (2018): 158-172.

[24]. T. Mitsuyuki, K. Hiekata, T. Goto, "Evaluation of project architecture in software development mixing waterfall and agile by using process simulation." Journal of Industrial Integration and Management, vol, 2-02 (2017): 1750007.

[25]. 満行泰河, 稗方和夫, 松原洸也, 大和裕幸, Bryan Moser, "船舶建造プロセスシミュレーションを用いた生産設備の導入に関する研究." 日本船舶海洋工学会論文集, vol. 24 (2016): 291-298.

[26]. 小林高之, 山口誠, 日比野浩典, "生産システム設計・改善時における生産性と消費エネルギ量のシミュレーションシステム." *日本機械学会論文集*, vol. 82-835 (2016): 15-000502.

[27]. 満行泰河, 稗方和夫, 大和裕幸, "電力ピークカットを考慮した作業計画立案手法の研究." *日本船舶海洋工学会論文集*, vol. 15 (2012): 123-126.

[28]. J. F. Maier, D. C. Wynn, W. Biedermann, U. Lindemann, P. J. Clarkson, "Simulating progressive iteration, rework and change propagation to prioritise design tasks." *Research in Engineering Design*, vol. 25-4 (2014): 283-307.

[29]. 満行泰河, 大和裕幸, 稗方和夫, モーザーブライアン, 磯沼大, 岡田伊策, 笈田佳彰. "システム開発プロジェクトにおける手戻りリスクを考慮したタスク優先ルール設計に関する研究." *日本機械学会論文集*, vol. 82-835 (2016): 15-00474.

[30]. T. Horii, Y. Jin, R. E. Levitt, "Modeling and analyzing cultural influences on project team performance." *Computational & Mathematical Organization Theory*, vol.10-4 (2005): 305-321.

[31]. Y. Suzuki, M. Yahyaei, Y. Jin, H. Koyama, G. Kanga, "Simulation based process design: Modeling and applications, *Advanced Engineering*

Informatics, vol. 26-4 (2012): 763-781.

[32]. 満行泰河, 大和裕幸, 稗方和夫, “複数設計業務のシミュレーションと組織内人員の最適配置” *日本機械学会論文集 C 編*, vol. 79-806 (2013): 3930-3938.

[33]. Enterprise Dynamics <http://www.incontrolsim.com/software/enterprise-dynamics> Accessed on: Jan 22nd, 2019.

[34]. E. Gelenbe, H. Guennouni, “FlexSim: A flexible manufacturing system simulator.” *European Journal of Operational Research,* vol. 53-2 (1991): 149-165.

[35]. 日比野浩典, “ICT を活用したものづくり：設備シミュレーション技術.” *精密工学会誌*, vol. 81-3 (2015): 230-232.

[36]. 貝原俊也, “IoT 環境下の「考える工場」実現を目指す実仮想融合型生産システム.” *計測と制御*, vol. 55-1 (2016): 53-58.

[37]. X. Zent, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, R. Ranjan, “IOTSim: A simulator for analyzing IoT applications.” *Journal of Systems Architecture*, vol. 72 (2017): 93-107.

[38]. Microsoft Azure IoT Solution Accelerators, <http://www.azureiotsolutions.com/Accelerators> Accessed on: Jan 22nd, 2019.

[39]. 森健一郎, “フィジカル・システムと連動した生産設備シミュレーション（設備シミュレーションを構築するためのコンポーネント標準化の提案）.” *日本機械学会論文集*, vol. 82-835 (2016): 15-00476.

[40]. T. F. Edgar, E. N. Pistikopoulos, “Smart manufacturing and energy systems.” *Computers and Chemical Engineering*, vol. 114 (2008): 130-144.

[41]. S. Masoud, Y. J. Son, C. Kubota, R. Transtad, “Evaluation of simulation-based optimization in grafting labor allocation.” *Applied engineering in Agriculture*, vol. 34-3 (2018): 479-489.

[42]. M. W. Merkhofer, “Quantifying judgmental uncertainty: Methodology, experiences, and insights.” *IEEE Transactions on Systems Management and Cybernetics*, vol. SMC-17-5 (1987): 741-752.

[43]. G. G. Shephard, C. W. Kerkwood, “Managing the judgmental probability

elicitation process: A case study of analyst/manager interaction." *IEEE Transactions on Engineering Mangement*, vol. 41-4 (1994): 414-425.

[44]. A. A. Yassine, D. E. Whitney, T. Zambito, "Assessment of rework probabilities for simulating product development processes using the design structure matrix (DSM)." *ASME International Design Engineering Technical Conferences* (2001).

[45]. R. G. Sargent, "Verification and validation of simulation models." *Proceedings of the 36th Winter Simulation Conference* (2004): 17-28.

謝辞

本研究を進めるにあたって、多くの方々にご助力を頂きました。ここに感謝の意を述べさせていただきます。

指導教員である東京大学大学院新領域創成科学研究科人間環境学専攻　准教授　稗方和夫先生には、修士課程の 2 年間、大変お世話になりました。他分野出身かつ研究経歴の浅い私に対して丁寧なご指導をしてくださったおかげて、一歩一歩着実に、研究を進めることができました。研究プロジェクトの関与や学会発表など、私にとっては、研究の推進ができるきっかけだけでなく、日本文化を深く知り、溶け込む機会でもあり、多くな成長をもらえることができました。また、稗方先生からは生活面でも多くな配慮をいただきまして、感謝の念に堪えません。ありがとうございました。

東京大学大学院新領域創成科学研究科人間環境学専攻　准教授　Bryan R Moser 先生には、研究に対して、多くな情報と励ましをいただきました。Moser 先生は、いつも研究に対してとても情熱的であり、私に大きな刺激を与えてくれました。また、国際交流の機会も多く設けてくださり、異文化コミュニケーションの難しさと面白さを、身をもって体験することができました。ありがとうございました。

横浜国立大学大学院工学研究院　システムの創生部門 准教授　満行泰河先生には、本研究の先行研究の著者として、大事なアドバイスを多くいただきました。満行先生には、修士 1 年の間、研究の方向性を示してくださった上、多くのご指導をいただきました。横浜国立大学への移籍後も、修士論文の細かいところについて、多忙の中、相談を乗って頂きました。満行先生からは、いつも沢山の優しさと元気を頂いています。修士 1 年の時に頂いた冷蔵庫も、大事に使わせて頂いてます。ありがとうございました。

東京大学大学院工学系研究科システム創成学専攻　技術専門員　榎本昌一様には、研究室のサーバなど、インフラに関する支援をいただきました。駅伝大会の時に示して頂いた若者に負けない精神も、運動嫌いな私に刺激を与えてくれました。ありがとうございました。

日本学術振興会外国人特別研究員　Giles Bruno Sioen 様には、研究の立ち位置や

フォーマットなどについて、沢山指摘をいただき、お陰様で、研究のロジックについて色々検討することができました。また、Sioen 様には、人間関係に対して不器用な私からの相談を、何度も乗っていただきました。頂いた言葉やアドバイスは、社会に踏み入れても、私の大事な糧となると思います。ありがとうございました。

秘書の山本和子様、大森容子様、大塚朋子様には、研究室の諸手続きなど、事務面のサポートを頂いており、研究に専念できる環境を作っていただきました。また日頃から、優しい励ましや気遣いも頂き、私の支えとなりました。ありがとうございました。

研究室の先輩である岡田伊策様、笈田佳彰様、和中真之介様、去年の卒業生である伊藤航大様、岡田航太様には、研究や日常生活で多くのご指導を頂きました。岡田様には、就職活動について、色々とアドバイスを頂きました。笈田様には、お忙しい中でも、修士論文の丁寧なレビューを頂き、細かいところまで見てくださいました。和中様には、研究面の丹念のご指導はもちろん、大学院生としての責任感についても、多くのご鞭撻を賜りました。伊藤様と岡田様には、真剣に研究を取り込む姿を示して頂き、薫陶を与えてくれました。ありがとうございました。

研究室の同輩である水林義博様、宇野健介様、石原祥太郎様には、研究室生活において、お互いに切磋琢磨することができました。水林様からは、研究について客観視の質問を頂きました。宇野様には、研究室の運営など、仕事面で大変お世話になり、沢山のご協力をいただきました。石原様には、ご在籍時にいつも怠らない姿勢を示してくださり、良い刺激となりました。ありがとうございました。

研究室の後輩である松尾康平様、笠原達也、Taskia Mst Khatun 様、趙之楠様、学部 4 年生である茶屋愛太郎様、堀井悠司様、去年の卒論生である小沢健悟様、三浦笑峰様には、研究生活でのご支援をいただきました。松尾様には、修士論文の日本語の校正を、素早くかつ極めて丁寧にしていただき、大変助けになりました。笠原様には、私の話し相手になっていただき、他愛もない話も聞いて頂きました。Taskia 様と趙様には、入学早々にも関わらず、研究室のお仕事を引き受けて頂きました。茶屋様と堀井様には、共に研究を取り込む仲間となり、お陰で苦のない研究生活を過ごすことができました。ありがとうございました。

留学生である Nat Hengsadeekul 様、秦圓圓様、劉雨泠様、Shekar shivhare 様には、研究生活を通して、国際的な視点と考え方を教わりました。ありがとうございました。

最後に、四六時中に私の大変さや喜びなど様々の感情を受け入れて頂いた友人、そして、金銭面の支援を含め、ずっと応援し続けてくださった家族にも、感謝を伝わせて頂きます。ありがとうございました。

Appendix

A. ケーススタディ 1 における実績データの生成

以下では、既存モデルに対して、簡単な説明を行った後、実績データを生成するための入力値を説明する。

既存モデルでの作業間の依存関係は、本研究の提案手法と同じく、DSM を用いて定義する。ただし、既存モデルの DSM 上では、タスク間の情報伝達も示している。作業はクリティカルパスと順位則ではなく、DSM の上から下の順で、依存関係を満足しているタスクを実行する。また、不確実性要素の入力値として、作業の最短期間、最頻期間、最長期間、学習による工期短縮時の比率（LC %, Learning Curve Effect）、手戻り確率と手戻り時の工期短縮比率がある。

仮想プロジェクトにおいて、Table 4-1 の DSM 以外に、実績データを生成するための入力値を Table 1、Table 2 と Table 3 に示す。

Table 1 仮想プロジェクトの実績データを生成するための入力値(1)

タスク名	最短期間	最頻期間	最長期間	LC %
A	2	3	4	80%
B	5	5	5	100%
C	10	10	15	50%

Table 2 仮想プロジェクトの実績データを生成するための入力値(2)：DSM2

タスク名	ID	1	2	3
A	1		0.4	0.3
B	2	1		0
C	3	1	0	

Table 3 仮想プロジェクトの実績データを生成するための入力値(3)：DSM3

タスク名	ID	1	2	3
A	1		0.5	0.5
B	2	0.5		0
C	3	0.3	0	

Table 1 では、各タスクの作業期間に関する入力値を示した。タスクの作業期間は、三角分布に従い、シミュレーションモデルの初期化の段階で生成される。最短期間、最頻期間、最長期間は、それぞれ、三角分布における最小値、最頻値、最大値と対応する。Table 1 の最後列は工期短縮時の比率である。既存モデルでは、人員の学習によって、同じタスクを再度実行する時に、作業期間が短縮される可能性があると仮定しているため、この比率を用いて、2 回目以降の作業期間の、元作業期間との比を示す。補足の説明として、ここでは、タスク B に対しては、手戻りのみを表現するため、最短期間、最頻期間と最長期間は、同じ値を与えた。

Table 2 では、タスク間の手戻り確率を示した。対角線以上の数値は、縦にある下流のタスクが完了した時点に、横にある上流のタスクへ手戻りする確率を示し、対角線以下の数値は、逆手戻りが発生した後に、手戻り先である上流タスクが、下流タスクに影響を与える確率を示す。また、既存モデルの手戻りに対する定義は、手戻り先のタスクが完了したか否か関わらず、一定の工期が追加されることである。ここでは、A タスクの再度実行は必ず下流のタスクに影響を与えると設定する。

Table 3 では、タスク間の手戻りによる影響率を示した。Table 3 は、Table 2 と対応するものであり、手戻り発生時に、手戻り先に対する追加工期を計算するための比率を示す。この比率は、手戻りによる追加工期と、横のタスクの元作業期間の比である。学習による工期短縮の仮定に加え、手戻りによる追加工期は、手戻り先のタスクの元作業期間、手戻りによる影響率、および工期短縮時の比率の三者の乗積である。

既存モデルの実行アルゴリズムを以下に示す:

1 三角分布に従い、各タスクの作業期間を決め、残り期間を作業期間と設定する。

2 タイムステップ毎に:

2.1 全てのタスクに「作業不可」の状態値を与える。

2.2 DSM の上位から順に探索し、依存関係を満足し、なお残り期間が 0 より大きいタス

クを、1つ取り出し、「作業中」の状態値を与える。

2.3 2.2 で取り出したタスクの次のタスクから、並行できる作業を探索する。依存関係を満足し、なお残り期間が 0 より大きいタスクが1つあれば、探索を停止し、該当タスクに「作業中」の状態値を与える。

2.4 状態値が「作業中」である全てのタスクに対して、以下の手順を行う:

2.4.1 残り期間を減らす。

2.4.2 残り期間<0.01 の場合、残り期間を 0 とする。

2.4.3 本タイムステップにおいて、2.4.2 の処理を経過したタスクに対して、タスクが対応する Table 4-3 の列において、対応する行の上にある数値を見る。1つの値につき、1個の 0 から1の乱数が生成される。乱数が手戻り確率より大きい場合、以下の処理を行う:

2.4.3.1 作業期間、LC%および Table 3 から読み込んだ数値によって、手戻りによる延長工期を計算し、手戻り先のタスクの残り期間に追加する。追加後、手戻り先のタスクの残り期間が元の作業期間より長い場合、残り期間を、元作業期間×0.9 とする。

2.4.3.2 手戻り先のタスクが対応する Table 2 の列において、対応する行の下にある数値を見る。1つの値につき、1個の 0 から1の乱数が生成される。乱数が手戻り確率より大きい場合、下流タスクに対して、2.4.3.1 と同じように、延長工期を残り期間に追加する。追加後、期間追加されたタスクの残り期間が元の作業期間より長い場合、残り期間を、元作業期間×0.9 とする。

3 全タスクの完了を判断する。全てのタスクの残り期間が 0 より小さい場合、シミュレーションを完了する。逆の場合、2 の諸手順を繰り返す。

B. ケーススタディ 2 における実績データの生成

ケーススタディ2 の実績データを生成するために、既存モデルに対する入力値を Table 4 から Table 6 に示す。これらのパラメータに基づいて、Appendix A を参照し、実績データを生成する。

Table 4 設計プロジェクトの実績データを生成するための入力値(1)[2]

タスク ID	最短期間（日）	最頻期間（日）	最長期間（日）	LC %
1	1.9	2	3	35%
2	4.75	5	8.75	20%
3	2.66	2.8	4.2	60%
4	9	10	12.5	33%
5	14.3	15	26.3	40%
6	9	10	11	100%
7	7.2	8	10	35%
8	4.75	5	8.75	100%
9	18	20	22	25%
10	9.5	10	17.5	50%
11	14.3	15	26.3	75%
12	13.5	15	18.8	30%
13	30	32.5	36	28%
15	4.5	5	6.25	70%

Table 5 設計プロジェクトの実績データを生成するための入力値(2)[2]

タスクID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1		0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.4		0	0	0	0	0	0	0.2	0	0	0	0	0
3	0	0.5		0.4	0	0	0	0	0	0	0	0	0	0
4	0.3	0	0.5		0	0	0	0	0	0	0	0	0	0
5	0.4	0	0.5	0		0.1	0	0.1	0	0	0	0.3	0.1	0
6	0.1	0	0	0	0.4		0	0	0	0	0	0	0	0
7	0.4	0	0	0	0	0.4		0	0	0	0	0.5	0	0
8	0	0	0	0	0	0.5	0		0	0	0	0	0	0
9	0.4	0	0.5	0.5	0	0	0	0.5		0	0	0	0	0
10	0	0	0	0.1	0	0.5	0.2	0.1	0		0.4	0	0	0
11	0	0	0	0	0	0.5	0.5	0.5	0	0.5		0	0	0
12	0.4	0	0	0	0	0.4	0.5	0	0	0.5	0.4		0	0
13	0.5	0	0	0	0.5	0	0	0	0	0	0	0.4		0
14	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	

Table 6 設計プロジェクトの実績データを生成するための入力値(3)[2]

タスクID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1		0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.5		0	0	0	0	0	0	0.1	0	0	0	0	0
3	0	0.3		0.5	0	0	0	0	0	0	0	0	0	0
4	0.4	0	0.8		0	0	0	0	0	0	0	0	0	0
5	0.1	0	0.1	0		0.1	0	0	0	0	0	0.3	0.1	0
6	0.1	0	0	0	0.3		0	0	0	0	0	0	0	0
7	0.5	0	0	0	0	0.8		0	0	0	0	0	0	0
8	0	0	0	0	0	0.5	0		0	0	0	0.5	0	0
9	0.3	0	0.3	0.3	0	0	0	0.3		0	0	0	0	0
10	0	0	0	0.1	0	0.5	0.4	0.3	0		0.3	0	0	0
11	0	0	0	0	0	0.5	0.5	0.3	0	0.3		0	0	0
12	0.5	0	0	0	0	0.3	0.5	0	0	0.5	0.5		0	0
13	0.9	0	0	0	0.9	0	0	0	0	0	0	0.3		0
14	0.5	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	