修 士 論 文

時間制約を考慮したソフトウェアプロセスの実行方式に関する研究

指導教官　片山卓也 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

藤井一郎

1997 年 2 月 14 日

目 次

第 1 章

はじめに

1.1　背景と目的

現在、社会が複雑化していく中、ソフトウェアの大規模、高品質化が求められている。しかし、ソフトウェアの開発は複雑かつ高度な技術を要求される思考中心の作業であり、従来のソフトウェア開発過程は、熟練のプロジェクトマネージャの頭の中だけに漠然とした状態で存在し明文化されていなかった。こうしたソフトウェア開発の課題を解決するために、ソフトウェア開発工程を形式的に記述し、実行するための方法であるソフトウェアプロセスに対する研究が活発化している。

ソフトウェアプロセスとは、ソフトウェア開発過程で行われる作業を明確に定義し、これを形式的に記述したものである。ソフトウェアプロセスを特定のプロセス記述言語で記述する技術はプロセスプログラミングと呼ばれる。実際の組織やプロジェクトの条件を付加した環境のもとで実行することをプロセス実行と呼ぶ。ソフトウェアプロセスを記述することにより、そのソフトウェア開発過程の性質や性能についての評価を行うことが可能になり、ソフトウェア開発の生産性の向上を図ることができる。

ソフトウェアプロセスを実際に実行するには資源を付加する必要がある。なぜなら、ソフトウェア開発は大量の資源を使用して行われるからである。資源を付加したモデルによってその資源とプロダクトあるいはアクティビティとの関係を示すことが可能となる。

そこで、本研究では資源を追加したモデルを提案し、そのモデルの実行方式を求めることを目的とする。

1.2 本論文の構成

本論文は以下のように構成される。

第 2 章では、ソフトウェアプロセスモデルのオブジェクト中心型プロセスとフェーズ中心型プロセスの定義とその役割を中心に述べる。

第 3 章では、オブジェクト中心型プロセスとフェーズ中心型プロセスの各モデルに対して時間制約を付加したモデルの拡張部分について述べる。また、作業員制約と要求制約の導入についても述べる。

第 4 章では、時間制約を含む実行可能グラフの作成について述べる。実行可能モデルを定義し、その作成手順を述べる。

第 5 章では、実行可能グラフがクライアントの要求する期間内に実行できないときの対処方法として、(1) 作業員の再割り当てによる方法、(2) チェックポイントの変更による PCP の変更と (3) 要求制約を用いた時間短縮の三つの方法によって全体時間を短縮する方法を述べる。

第 6 章では、今回提案した資源付きモデルの評価について述べる。

第 7 章では、まとめと今後の課題を述べる。

第 2 章

ソフトウェアプロセスの記述とその特徴

2.1 ソフトウェアプロセスの記述に対する要求

ソフトウェアプロセスモデルとは、ソフトウェアプロセスの構造に関する概略的記述のことをさし、プロセス記述言語とは、個々のプロセスを記述するために用いられる言語のことであり、ここ数年で多くの研究がされ、提案されている記述言語の数も多数ある。

ソフトウェアプロセスを記述するモデルにはいくつか要求されることがある。以下にそれをあげる。[2] [3] [7]

- 表現能力
  モデルの基本的な能力であり、プロダクトとアクティビティの関係、アクティビティの実行順序、実世界との関係 (スケジューリング、予算や作業員の割り当て／管理など) が記述できるか。

- 分析／検証、シミュレーション能力
  プロセスの進行に従ってモニタリングを行い、実行結果の分析や検証が可能か。また、新しいプロセスに対してシミュレーションにより結果の予測が可能か。

- 適用／進化の容易さ
  プロセスは実行中に常に変化するが、このような動的な変更に対し記述はどの程度有効か。進化のガイドラインを示すことが可能か。

- 理解のしやすさ
  大規模なプロセスにおいても理解が容易であるかどうか。

2.2 ソフトウェアプロセスの記述方式

2.2.1 プロダクトに着目した方法

望月ら [5] が現実のソフトウェアプロセスの分析を通じて、オブジェクト中心型プロセスとフェーズ中心型プロセスという 2 つのプロセスモデルに対する視点を与えた。プロダクトに着目したモデルがオブジェクト中心型プロセスであり、このモデルはオブジェクト間の静的な関係に注目し、さまざまなソフトウェア開発に適応できる普遍的なモデルである。

2.2.2 アクティビティの順序に着目した方法

S.Sutton ら [4]によって提唱された手続き型言語 APPL/A を用いて、対象となるアクティビティとその挙動を記述するモデルがある。

望月らのフェーズ中心型プロセスはアクティビティの順序に着目したモデルである。このモデルは、ソフトウェアプロセスをアクティビティの逐次あるいは並列実行の組合せから構成されるフェーズの繋りと考え、具体的な実行順序を決定するのに利用されるモデルである。

2.3 オブジェクト中心型プロセスとフェーズ中心型プロセス

2.3.1 概要

2 つのプロセスモデル、オブジェクト中心型プロセスとフェーズ中心型プロセスについてその定義と役割について [6]をもとに述べていく。**オブジェクト中心型プロセス (OCP)** とは、ソフトウェアの開発で作成される生成物であるオブジェクトの静的な関係に着目したモデルであり、**フェーズ中心型プロセス (PCP)** とは、オブジェクトを段階的に作成していく過程であるフェーズを記述したモデルである。

以下に OCP と PCP の特徴を述べる。

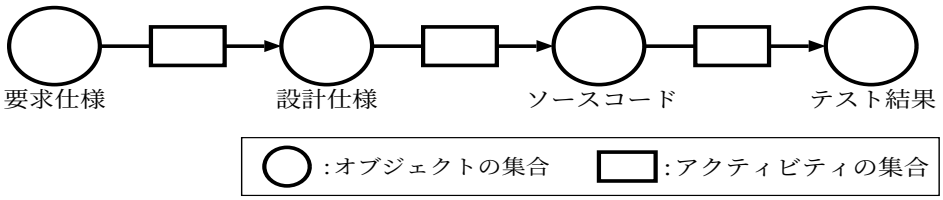

図 2.1: オブジェクト中心型プロセス図

オブジェクト中心型プロセス (OCP)

まず、オブジェクト中心型プロセスの中心となるオブジェクトの定義をする。**オブジェクト**とは、ソフトウェアプロセスの実行過程で作成、または修正される生成物のことをいい、設計仕様書、プログラムといったものがオブジェクトにあたる。また、オブジェクト間の関係を成立させるためのプロセス主体を**プロセス**、その構成要素を**アクティビティ**という。このアクティビティはあるオブジェクトからオブジェクトを生成するときの作業内容を示したものである。

オブジェクト中心型プロセス (OCP) とは、オブジェクトの集合とアクティビティの集合との関係からソフトウェアプロセスを記述したものである。その概念図を図 2. 1に示す。

フェーズ中心型プロセス (PCP)

OCP はオブジェクトの静的な関係、つまり要求仕様から設計仕様へというような関係を理解するには有用であるが、これらのオブジェクトを実際にどのような順序で作成していくのかを示してはいない。実際のソフトウェア開発においては、設計仕様を例にとると、基本設計、概要設計、詳細設計のように段階的に中間オブジェクトを作成しながら、最終的な設計仕様書を作成していく。

そこで、実際にオブジェクトの作成の順序を示すモデルが必要になる。そのオブジェクトの作成手順を含んだモデルとしてフェーズ中心型プロセスがある。

フェーズとはある段階のオブジェクトを作成するためのアクティビティの集合であるとする。つまり、オブジェクトとして要求仕様と設計仕様があったとすると、それぞれ基本、概要、詳細の 3 段階から構成されているとき、要求仕様から設計仕様を作成するためのアクティビティも 3 段階から構成されると考えられる。**フェーズ中心型プロセス** (PCP) と

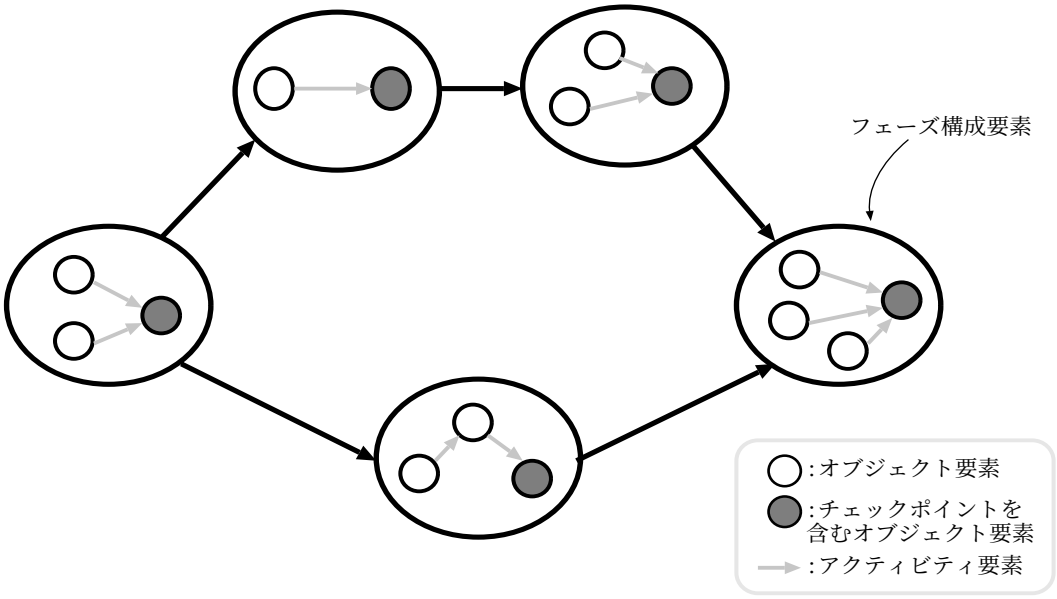

図 2.2: フェーズ中心型プロセス図

は、フェーズとその実行シーケンスを表現したものである。

PCP の概念図を図 2.2に示す。

2.3.2 OCP、RCP の形式的定義

PCP を構成するには、オブジェクトの段階を決定し、アクティビティをフェーズ毎に分ける必要がある。オブジェクトの段階はオブジェクトの完成の度合を表すものであり、フェーズはアクティビティ実行の単位と考えられる。

オブジェクトの段階の定義方法として、以下のような決定法を用いる。

1. オブジェクトの構造を分析し、各オブジェクトに対してオブジェクト構造を導入する。
2. オブジェクト構造に基づいてアクティビティを各オブジェクト要素を生成するための段階に定義し、各段階のオブジェクトを作成するためのアクティビティ要素の集まり (フェーズ構成要素) を決定する。
3. フェーズ構成要素および実行シーケンスを依存関係に従って結合することにより、全体の実行シーケンス (=PCP) を得る

以上の方法に従って OCP から PCP を求める。

オブジェクト中心型プロセス

オブジェクト中心型プロセスを構成する方法を以下に示す。

1. 生成物とその間の関係を抽出する。
2. 同一の働きを持つ生成物であるが、その詳細度の段階が異なるものを一括してオブジェクトの集合を定義する。

OCP を形式的に記述すると以下のようになる。

定義 2.1

$OCP = (O, A)$

但し、O :オブジェクトの集合

A :アクティビティの集合

$A \subseteq O \times O$

オブジェクトグラフ

オブジェクトグラフとは、構成要素 (**オブジェクト要素**) を頂点として、辺が詳細化、あるいは部品化の関係を表す無循環な有向グラフのことである。ここでいう「詳細化」とは、各オブジェクト要素間には、基本設計→ 詳細設計のように、「Bは A を詳細化して得られる」という関係であり、「部品化」とは、設計仕様書のインターフェース設計、機能設計などの部品から構成されるといったような「Bは A の部品である」という関係のことをいう。

ここで、今後説明に利用する例について説明する。OCP において「要求仕様」から「設計仕様」に関する部分を用いることにする。オブジェクトの集合「要求仕様」では、オブジェクト要素として、要求仕様 RS と機能についての仕様 FN がある。また、オブジェクトの集合「設計仕様」は、オブジェクト要素として、概要設計 OV、インターフェイス設計 IF、機能設計 FS、各機能の詳細設計 F1,F2、インターフェイス設計 I F、機能の詳細設計 F1 との結合 CO、統合方式設計 I NT の 7 つのオブジェクト要素からなる。オブジェク

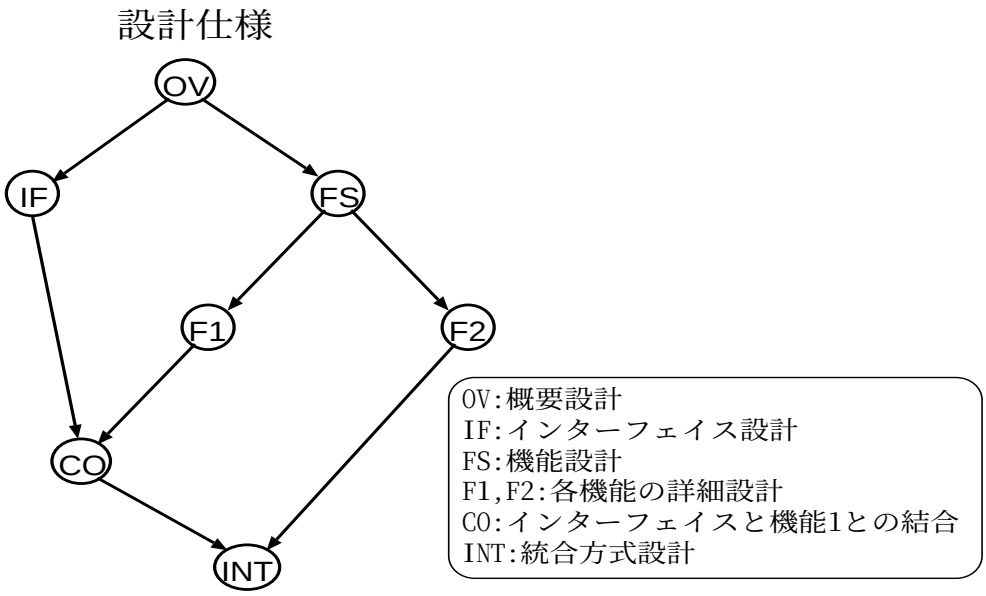

図 2.3: オブジェクトグラフ図 (設計仕様)

トの集合「設計仕様」に対するオブジェクトグラフを図 2. 3 に示す。オブジェクト同士の関連であるアクティビティは図 2. 4に示す通りである。

オブジェクトグラフを形式的に記述すると以下のようになる。

定義 2.2

$OG = (OE, R)$

但し、OE :オブジェクト要素の集合

R :オブジェクト要素間の関係

$R \subseteq OE \times OE$

$R^0 \stackrel{\text{def}}{=} \{(x, x) \mid \forall x \in OE\}$

$R^{i+1} \stackrel{\text{def}}{=} \{(x\ ,y) \mid \exists z\ ;\ (x\ ,z) \in R\ ,\ (z, y) \in R^i\}$

$R^+ \stackrel{\text{def}}{=} \bigcup_{i \geq 1} R^i$

$R^* \stackrel{\text{def}}{=} \bigcup_{i \geq 0} R^i$

但し、Rは非循環であるから、以下の条件を満たす必要がある。

条件 2.1

$\forall x\ ;\ (x,x) \notin R^{+}$

OCP と各 $o \in O$に対してオブジェクトグラフが与えられたとき、オブジェクトの構造は以下のように定義される。

定義 2. 3

$OS \stackrel{\text{def}}{=} (OG, Obj)$

但し、OG : オブジェクトグラフの集合

$Obj:\ O \rightarrow OG$

各オブジェクトにオブジェクトグラフを対応させる関数

$Obj(o) = OG \quad (o \in O)$

アクティビティの構造

アクティビティの目的は、オブジェクトの作成である。オブジェクトを細分化することによって、アクティビティもそれに対応し細分化される。オブジェクト要素は以下の 2 種類のアクティビティによって生成される。

1. 他のオブジェクトの集合の構成要素との関連を実現するもの。
   - OCP におけるアクティビティの依存関係から決まる。
2. 同じオブジェクトの集合の構成要素を詳細化するもの。
   - オブジェクトグラフにおける部品化、詳細化関係から決まる。

個々のオブジェクト要素を作成するためのアクティビティの部品を**アクティビティ要素**と呼ぶ。オブジェクト中心型プロセス $OCP = (O, A)$ とオブジェクト構造 $OS = (OG, Obj)$ が与えられたとき、アクティビティ要素 AE は依存関係 AD と詳細化関係 AR から構成され、以下のように与えられる。

定義 2.4

$$AE = AD \cup AR \quad \subseteq OE \times OE$$

但し、$AD \subseteq \{(x, y) \mid (e, f) \in A\ ,\ x \in OE_e\ ,\ y \in OE_f\}$

$AR \subseteq \{(x\ ,y) \mid (x\ ,y) \in R_e\ ,\ x \in OE_e\ ,\ y \in OE_e\}$

AE : アクティビティ要素

AD : 依存関係

AR : 詳細化関係

拡張 OCP[EOCP]

$OCP = (O, A)$ に各オブジェクトに対するオブジェクトグラフの集合 OG と対応するアクティビティ要素 $AE = AD \cup AR$を与えることで、構造を持った OCP(**拡張 OCP**) を以下のように定義できる。

定義 2．5

$$EOCP\ =\ (O\ , A,\ OG,\ Obj,\ AE,\ Acts)$$

$$Act\ s\ OE \rightarrow (OE \times OE)$$

$$Act\ (s e)\ =\ \{(x\ ,y) \mid (x\ ,y) \in AE\ ,\ y = o\ e\}$$

$Act\ (s e)$ はオブジェクト要素 $o\ e$を作成するために必要なアクティビティ要素の集合を与える関数である。

オブジェクトの集合「要求仕様」と「設計仕様」に対するオブジェクトグラフおよびアクティビティ要素を図 2.4に示す。

オブジェクト ステージ

ソフトウェア開発プロジェクトでは、生成されたオブジェクトに対して途中で段階的に検査を行うことによって、早い段階で操作の誤りの発見し、手戻りの必要性を最小限にするための箇所がいくつか設けられている。オブジェクト要素の一部に検査のための条件が付随していて、オブジェクト要素がその条件を満たしたときに初めてその段階のオブジェクトは完成したものと考えられる。

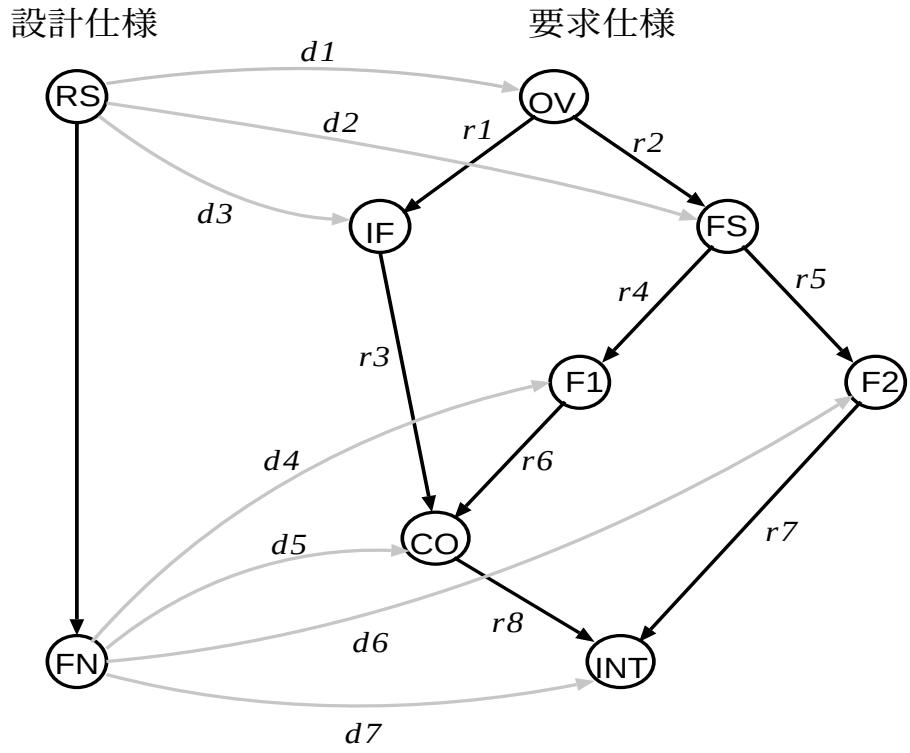

図 2.4: 拡張オブジェクト中心型プロセス図

検査条件が付加されているオブジェクト要素を**チェックポイント**と呼ぶ。あるオブジェクト $e \in O$に対するオブジェクトグラフを $OG_e = (OE_e, R_e)$ とする。オブジェクト e におけるチェックポイントの集合 $CP \subseteq OE_e$が与えられると、各検査段階でのオブジェクトの集合 S(**オブジェクト ステージ**) は以下のように定義できる。

定義 2.6

$$S \stackrel{\text{def}}{=} \bigcup_{\forall cp \in CP} \{(oe, r)\}$$

但し、$oe = \{x \mid x \in OE_e,\ (x, cp) \in R_e^*\}$

$$r = \{(x, y) \mid x, y \in oe,\ (x, y) \in R_e\}$$

オブジェクトステージの各要素 $s = (oe, r)$ は、該当するチェックポイントにおけるオブジェクト要素を生成するために必要なオブジェクト要素を全て含む部分グラフである。オブジェクトステージの要素 $s_i = (oe_i, r_i), s_j = (oe_j, r_j)$ に対して、半順序関係 $\sqsubseteq$ を定義す

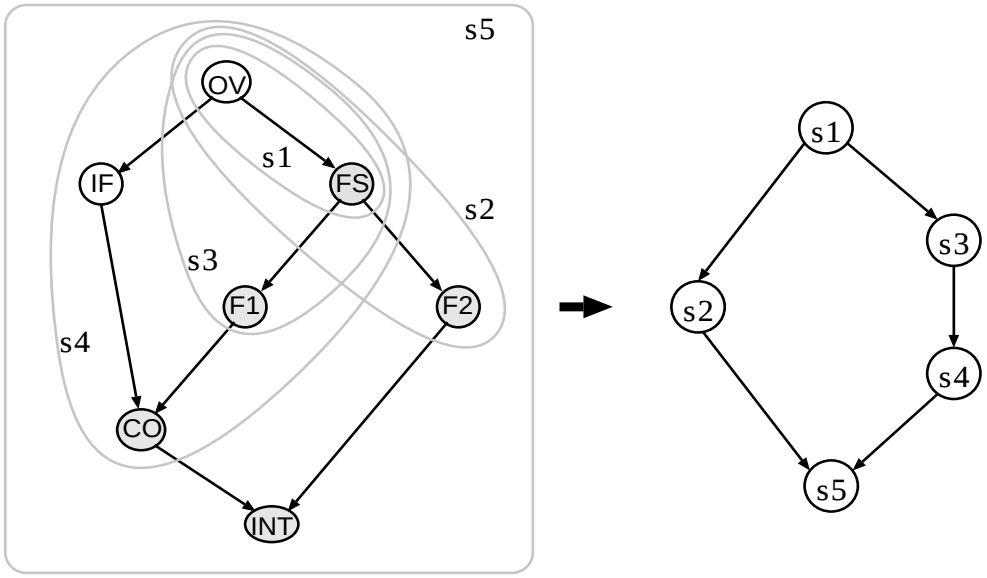

図 2.5: ステージグラフ図

ると、$(S, \sqsubseteq)$ は有向グラフを構成する。これを**ステージグラフ**と呼ぶ。

定義 2.7

$$s_i \sqsubseteq s_j \stackrel{\text{def}}{=} oe_i \subseteq oe_j \ \wedge \ \forall (oe_k, r_k) \in S \ ; \ oe_i \not\subseteq oe_k \,/\, \not\in e_j$$

オブジェクトの集合「設計仕様」に対するステージグラフを図 2.5に示す。灰色で塗りつぶしてあるオブジェクト要素は検査条件が付加されたオブジェクト要素であるチェックポイントを示している。オブジェクトステージ Sは $\{s_1, s_2, s_3, s_4, s_5\}$ であり、半順序関係 $\sqsubseteq$ は $\{(s_1, s_2), (s_1, s_3), (s_3, s_4), (s_2, s_5), (s_4, s_5)\}$ である。

フェーズ構成要素

フェーズ構成要素とは、差分オブジェクトを作成するためのアクティビティ要素の集合である。

定義 2.8

$$Pe(s) \stackrel{\text{def}}{=} \bigcup_{y \in DS} Acts(y)$$

但し、$DS = \bigcup_{s' \in S, s' \sqsubseteq s} (oe' - oe)\ ,\ s = (oe, r), s' = (oe', r')$

DS : それ以前のチェックポイントにおけるオブジェクトとの差分

Pe : フェーズ構成要素を求める関数

図 2.4の EOCP におけるフェーズ構成要素は以下の通りである。

$$\begin{aligned} Pe(s_1) &= \{d1, d2, r2\} \\ Pe(s_2) &= \{d6, r5\} \\ Pe(s_3) &= \{d4, r4\} \\ Pe(s_4) &= \{d3, d5, r1, r3, r6\} \\ Pe(s_5) &= \{d7, r7, r8\} \end{aligned}$$

フェーズ中心型プロセス (PCP)

あるチェックポイント $cp \in CP$におけるオブジェクトステージの要素を s とする。フェーズ構成要素 $Pe(s)$、および各チェックポイントにおいて検査を行うためのアクティビティ Cが与えられたとき、各フェーズ構成要素の内容は以下の Seqで定義され正規言語で表現できる。

定義 2 . 9

$$Seq(cp) \stackrel{\text{def}}{=} (AC)+$$

但し、$A : Pe(s)$

$C : CP$における検査アクティビティ

プロセス全体の実行シーケンスは、各チェックポイントにおけるオブジェクトを作成するための実行シーケンスを、依存関係に矛盾しないようにプロセス全体に渡って結合したものである。

オブジェクト中心型プロセス $OCP = (O, A)$、オブジェクト構造 $OS = (OG, Obj)$、アクティビティ要素 AE 、および各オブジェクトに対するチェックポイント集合 CPが与え

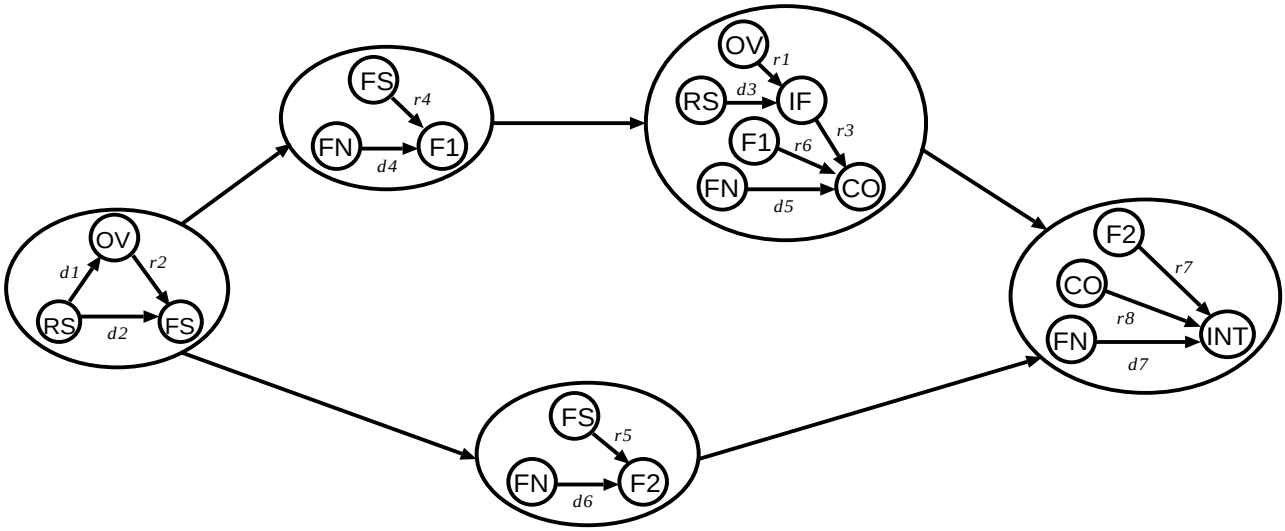

図 2.6: フェーズ中心型プロセス図

られたとき、これまでの定義を使用して正規表現を頂点とする有向グラフを**フェーズ中心型プロセス**とする。以下に、その定義を記す。

定義 2.10

$PCP \stackrel{\text{def}}{=} (S, PE, SEQ)$

但し、S : 全てのオブジェクトに対するオブジェクトステージの集合

PE : 全てのオブジェクトに対するフェーズ構成要素の集合

SEQ : 全てのチェックポイントにおける実行シーケンスの集合

有向グラフの全ての頂点を一回ずつ通るようなパスは、最終的なオブジェクを生成するためのアクティビティの実行順序となる。

図 26 は図 25 の PCP 図である。

第 3 章

時間制約付きモデルへの拡張

3.1　ソフトウェアプロセスにおける資源の扱いについて

ソフトウェアプロセスを実際に実行するためには資源を必要とする。なぜなら、ソフトウェア開発をする作業を行うのは人間であり、その作業には時間や機器など資源を使用するからである。

ここでは種々の資源の中から作業員と時間に着目する。その理由として、ソフトウェアの開発を行う主体は人間であり、またすべての活動には時間の概念が付随するからである。すなわち、ソフトウェア開発においては常に作業員と時間に関する要求を考慮する必要がある。ソフトウェア開発における資源の中には機器やコストといったものもあるが、これらの資源は作業員や時間に密接に関係している。例えば、機器は主に作業員によって使用されるので、これを作業員へのパラメータとして考えることによって実行モデルに反映することが可能となる。そのため、ここではこれらの資源については述べない。

最初にソフトウェア開発における作業員と時間について定義を述べる。

ソフトウェア開発における**作業員**とは、ソフトウェアプロセスの末端において実際に作業を行う人を指す。作業員はその役割や責任において、プロジェクトを管理するプロジェクトマネージャ、設計を担当する設計者、コーディングを担当するプログラマなどの人に分類される。プロジェクトの規模により作業員の人数は異なる。例えば、大規模なプロジェクトでは設計者、プログラマは複数必要であるが、小規模なプロジェクトでは、設計者がプログラマを兼任するといった場合もありうる。また、作業員は各々異なった経験や技術などを身に付けており、その経験や技術によってプロジェクトの進行や生成物の品質

に大きな影響を与える。

ソフトウェア開発における**時間**には、プロジェクトの開始時間やその終了時間、個々の作業に対する作業時間、作業員が 1 日に作業に当たれる時間など、さまざまな側面がある。ソフトウェアプロセス記述・実行モデルはこれら複数の側面を扱えることが要求される。

次節では実行モデルにおいて考慮すべき時間制約と作業員制約について述べ、3 節では時間制約に影響を与える要求制約について述べる。

3.2 時間制約の導入

ソフトウェア開発のプロジェクト管理においては、決められた期間 (納期) 内に限られた資源をいかに効率よく配分し、高品質の生成物を作り出すかということが最重要である。

ソフトウェア開発プロジェクトを納期内に終えるためには、プロジェクトの計画時に各アクティビティ要素の時間配分を十分に注意して行う必要がある。ここで、個々のアクティビティ要素の実行に必要な時間を正確に見積もることができるかが、プロジェクトの成功に大きく影響する。もし、実際の作業時間よりもはるかに少なく時間を見積もると、その後のすべての作業に影響が波及し、プロジェクト全体の遅れや品質の低下につながる。また、実際のプロジェクトの実行中には予測もつかないことがしばしば起きる。例えば、ソフトウェア開発に必要な機器にトラブルが発生してその機器を使用する作業が行えなくなる、あるいは作業員がなんらかのトラブルに巻き込まれてプロジェクトに参加できなくなるなどである。このような突発的事態に対応することが、プロジェクトの管理には要求される。

ここでは、時間制約を形式的に扱うことが可能な実行モデルを PCP を拡張することで定義する。このモデルでは、各作業の実行に必要な時間量と全体に必要な時間量が形式的に定義される。

3.2.1 要求時間の見積り方法

まず、ある作業を行うためにはどのくらいの時間が必要かという時間量のことを**要求時間**と呼ぶことにする。各アクティビティ要素の要求時間の見積り方法としては、基本的にプロジェクトを管理するプロジェクトマネージャとプロジェクトに関わる作業員が過去に

行ったことのあるプロジェクトからの経験や、各オブジェクト要素の規模や質から必要と思われる作業時間を算出する。つまり、アクティビティ要素の要求時間はプロジェクトマネージャや作業員が経験から見積もった値ということになる。このような経験値である要求時間は非常にあいまいな値であるが、過去に同様なソフトウェア開発を行ったことのあるプロジェクトマネージャや作業員であれば、実際の作業に必要な時間に近い要求時間を見積もることができると考えられる。しかし、初めて関わる種類のソフトウェア開発においては、この要求時間を見積もることが難しい。

このようなあいまいな時間を見積もる方法として統計を使用した「3 点見積り法」[10] という方法がある。この「3 点見積り法」とは、3 つの値、つまり最早時間 ft、標準時間 st、最遅時間 lt の値を使って以下の式より、要求時間を算出する方法である。

$$\text{要求時間} \ = \ \frac{ft \ + \ 4st \ + \ lt}{6}$$

最早時間とは、あるアクティビティ要素を実行するときに最良の条件下で実行できる場合の時間であり、最遅時間とは最悪の条件下で実行しなくてはならない場合の時間のことをいう。標準時間は、両者の中間で標準的な条件の下での所要時間である。

この方法によって求められる要求時間は、正確に実際の作業時間を見積もることができるとはいえないが、はっきりとした要求時間を求めることが難しいときには、基準となる時間を導くことができると期待される。

3.2.2 アクティビティ要素の要求時間について

最初に、各アクティビティ要素 ae の要求時間をモデル化する。まず、アクティビティ要素から要求時間を求める関数を RT とする。RT を形式的に記述すると以下のようになる。

定義 3.1

$$RT: \ AE \rightarrow T$$
$$RT = \{(ae, \ t) \mid ae \ \in AE \ , \ t \in T\}$$

T は時間 (実数) の集合を示している。$RT(ae)$ は各アクティビティ要素 ae の要求時間を示す。アクティビティ要素からその要求時間を求める方法は、前に述べたようにプロジェクトマネージャや作業員による経験を基に決定される値を用いる。

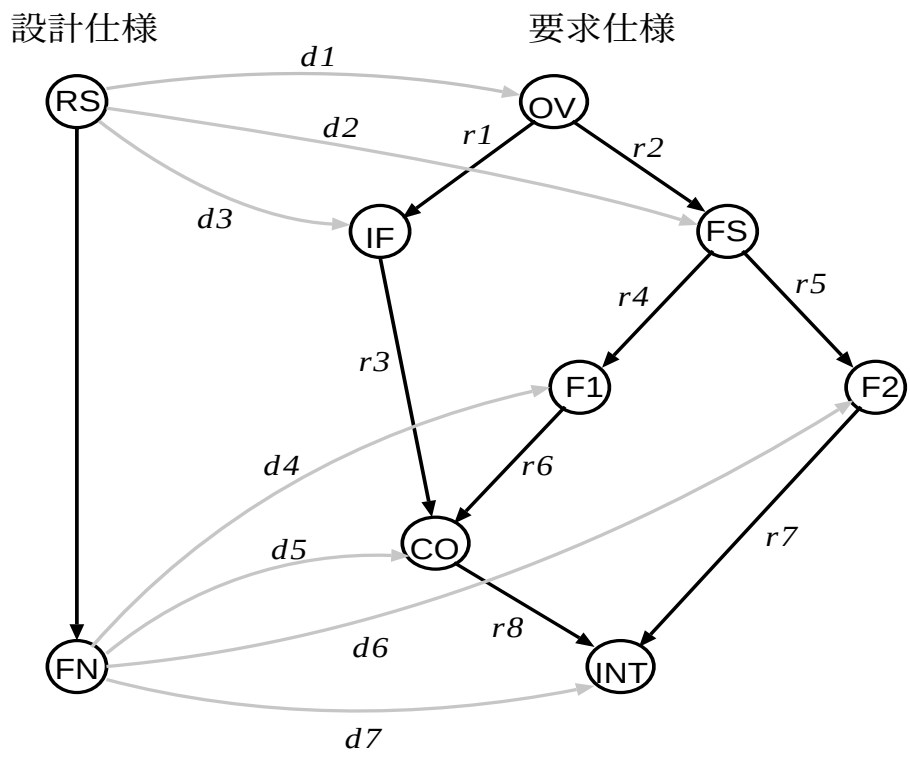

図 3.1: 拡張オブジェクト中心型プロセス図

例として第 2 章で述べた拡張 OCP グラフを図 3. 1に示し、各アクティビティ要素とその要求時間との対応表を表 3. 1に示す。

3.2.3 作業員に関する制約

ここで、作業員に関する制約について述べておく。個々の作業員は様々な技能や経験を有している。そのため、ある作業員は他の作業員に比べて特に優れていたり、ある作業に対しては特定の作業員にしか作業ができないような場合が考えられる。このような優れた作業員や特殊な作業を行える作業員はフェーズ構成要素に優先的に割り当てることが望まれる。

次に、PCP への作業員の割り当てに関する制約を示す。

1. PCP 上でフェーズ構成要素が並行実行している部分には、同一の作業員を並行に作業をさせないように割り当てる。

ae	$d1$	$d2$	$d3$	$d4$	$d5$	$d6$	$d7$	$r1$	$r2$	$r3$	$r4$	$r5$	$r6$	$r7$	$r8$
$RT(ae)$	t_{d1}	t_{d2}	t_{d3}	t_{d4}	t_{d5}	t_{d6}	t_{d7}	t_{r1}	t_{r2}	t_{r3}	t_{r4}	t_{r5}	t_{r6}	t_{r7}	t_{r8}

表 3.1: アクティビティとその要求時間の対応表

これは一つのフェーズ構成要素には一人の作業員が割り当てられるので、並行に実行できる複数のフェーズ構成要素に同一の作業員を割り当てると、並行に実行することができなくなるためである。

2. スキルを考慮して作業員に割り当てる。

   これは、特定の作業に秀でた作業員や特殊な機器を使用できる作業員に対しては、その作業に優先的に割り当てることにより、他の作業員に作業させるより効率がよく、教育する時間がいらないためである。

プロジェクトに参加する作業員の集合を DEV、フェーズ構成要素の集合を PEとしたとき、フェーズ構成要素に作業員を割り当てる関数 AGTを、形式的に記述すると以下のようになる。

定義 3.2

$$AGT : PE \rightarrow DEV$$
$$AGT = \{(pe, a) \mid pe \in PE,\ a \in DEV\}$$

関数 AGT はフェーズ構成要素の制約と作業員の制約を満たすように作業員を割り当てる。ここでいうフェーズ構成要素の制約とは、フェーズ構成要素を実行するのに必要な技術力、使用言語などが挙げられる。例えば、プログラマ $progA$, $progB$の二人がいるとすると、プログラマ $progA$ はプログラム歴 10 年、使用できるプログラミング言語はFORTRAN,C である。一方、プログラマ $progB$はプログラム歴 2 年、使用できるプログラミング言語は Lisp, MLであるとしたとき、難易度の高い C 言語のソースコードを記述する必要のあるフェーズ構成要素には、作業員とフェーズ構成要素の制約より、プログラマ $progA$ を割り当てるように関数 AGT を定義する必要がある。

PE	pe1	pe2	pe3	pe4	pe5
$PT(pe)$	pt_{pe1}	pt_{pe2}	pt_{pe3}	pt_{pe4}	pt_{pe5}

表 3.2: フェーズ構成要素とその要求時間の対応表

3.2.4 フェーズ構成要素の要求時間について

フェーズ構成要素はチェックポイントにおけるオブジェクト要素を生成するために必要なアクティビティ要素の集合である。各フェーズ構成要素に割り当てられる作業員を一人に制限することにより、フェーズ構成要素中のアクティビティの実行は逐次実行になる。そのため、フェーズ構成要素の実行に必要な要求時間はフェーズ構成要素中のアクティビティ要素の要求時間の総和に等しい。

まず、フェーズ構成要素 pe 中の全アクティビティ要素の要求時間を求める関数 STを形式的に記述する。STは各アクティビティ要素の要求時間を求める関数 RTのドメインをあるフェーズ構成要素 pe に制限したものであり、以下のように記述できる。

定義 3.3

$$ST(pe) = RT \lhd pe$$

関数 STによって求められるフェーズ構成要素 pe 中のすべてのアクティビティ要素の要求時間を用いて、フェーズ構成要素の要求時間を求める関数 PTを定義する。

定義 3. 4

$$PT : \ PE \rightarrow T$$
$$PT \ = \ \{(pe, pt) \mid pt = \sum_{(ae,t) \in ST(pe)} t\}$$

$PT(pe)$ はフェーズ構成要素 pe の要求時間を示す。

図 3.1の各フェーズ構成要素に対する要求時間を表 3.2に示す。表 3.2のフェーズ構成要素の要求時間を計算したものを以下に示す。

$$
\begin{aligned}
pt_{\mathsf{pe1}} &= t_{d1} + t_{d2} + t_{r2} \\
pt_{\mathsf{pe2}} &= t_{d6} + t_{r5} \\
pt_{\mathsf{pe3}} &= t_{d4} + t_{r4} \\
pt_{\mathsf{pe4}} &= t_{d3} + t_{d5} + t_{r1} + t_{r3} + t_{r6} \\
pt_{\mathsf{pe5}} &= t_{d7} + t_{r7} + t_{r8}
\end{aligned}
$$

3.3 要求制約の導入

3.3.1 オブジェクト要素に対する要求について

プロダクトは、クライアントから要求されている性能や機能を含むものでなくてはならない。高品質のプロダクトとはクライアントの要求した性質や機能といったものをすべて含み、かつバグのない生成物のことであるが、限られた資源量の中でこのような高品質のプロダクトを作成するのは一般的に困難である。一定の期間内に十分な品質をもったプロダクトを完成させることが困難な場合には、プロダクトの品質に対するクライアントの要求を低下、つまり機能の削除や性能の低下を行うことにより、開発プロジェクト全体が期間内に収まるようにする必要がある。このトレードオフを決定するための制約条件を**要求制約**と呼ぶ。

オブジェクト要素に対して要求の低下の度合を示すために**劣化係数**を導入する。劣化係数はプロジェクト内のすべてのオブジェクト要素に付加されるものとする。DEG をオブジェクト要素からその要求係数を求める関数とし、以下のように記述する。

定義 3.5

$$DEG:\ OE \rightarrow [0,1]$$

この劣化係数 $DEG(oe)$ は $0 \leq DEG(oe) \leq 1$ の範囲にあり、$DEG(oe) = 1$ のときはオブジェクト要素に対して要求された条件をすべて充足するようなプロダクトを作成し、$DEG(oe) = 0$ のときは要求条件を一つも充足しない、すなわちオブジェクト要素を全く作成しないことを意味する。

劣化係数の変更の仕方にはさまざまな方法が考えられるが、通常のソフトウェア開発では要求条件の変更 (機能の削除など) はクライアントとの話合いによる仕様の変更を必要

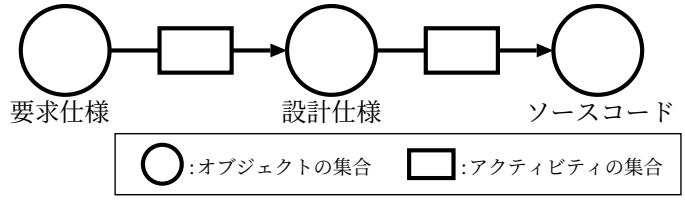

図 3.2: オブジェクト中心型プロセス図

とする。また、ソフトウェアの性能や機能といったものは要求仕様の時点でほぼ決定している。これらのことから、図 3. 2のような OCP 図においてオブジェクトの集合「要求仕様」、「設計仕様」、「ソースコード」があるとすると、オブジェクトの集合「要求仕様」のオブジェクト要素から後続のオブジェクトの集合「設計仕様」や「ソースコード」のオブジェクト要素へ伝播させることによって、オブジェクト要素の劣化係数を変更することが可能である。

次に、各オブジェクト要素の劣化係数を求める関数 $Degrade$ を形式的に記述する。その準備として、対象オブジェクト要素と先行オブジェクト要素の定義をする。**対象オブジェクト要素**とは、着目しているオブジェクト要素のことであり、**先行オブジェクト要素**とは、対象オブジェクト要素を生成するときに必要とするオブジェクト要素のことである。以下に先行オブジェクト要素を求める関数 $Preo$ を形式的に記述したものを示す。

定義 3.6

$$Preo(oe) = \{oe' \mid (oe', oe) \in AE\}$$

但し、oe: 対象オブジェクト要素

oe': 先行オブジェクト要素

図 3.3は対象オブジェクト要素と先行オブジェクト要素の関係を示している。オブジェクト要素 oe1, oe2, oe3 があり、oe3 を対象オブジェクト要素とすると oe1, oe2 が先行オブジェクト要素である。

$Degrade$ 関数は、対象オブジェクト要素 oe の劣化係数 $DEG(oe)$ を先行オブジェクト要素の集合 $Preo(oe)$ の劣化係数から求める関数である。

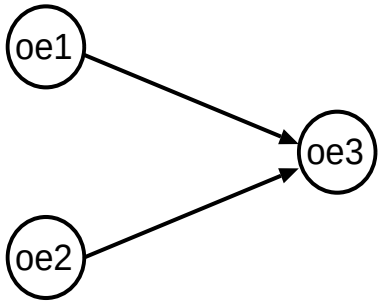

図 3.3: 対象オブジェクト要素と先行オブジェクト要素図

OE	RS	FN	OV	IF	FS	F1	F2	CO	I NT
$DEG(oe)$	q_{RS}	q_{FN}	q_{OV}	q_{IF}	q_{FS}	q_{F1}	q_{F2}	q_{CO}	$q_{\mathsf{I\ NT}}$

表 3. 3: オブジェクト要素とその要求係数との対応表

定義 3.7

$$Degrade: \ 2^{[0,1]} \rightarrow [0,1]$$
$$Degrad(\{DEG(Pre(oe))\}) = DEG(oe)$$

表 3. 3に図 3. 1のオブジェクト要素とその劣化係数の対応表を示す。q_{oe}は $DEG(oe)$ によって求められた劣化係数を意味する。各劣化係数はオブジェクトの集合「要求仕様」のオブジェクト要素 RS の劣化係数 $REQ(\mathsf{RS})$ を最初に与えることによって、オブジェクト要素の依存関係に従って各オブジェクト要素の劣化係数を変更していく。例えば、オブジェクト要素 OV の劣化係数 q_{OV}を求めるには、先行オブジェクト要素である RS の劣化係数 q_{RS}から求める。また、オブジェクト要素 I NTの劣化係数 $q_{\mathsf{I\ NT}}$は、その先行オブジェクト要素の F2 ,CO, FNの劣化係数 $q_{\mathsf{F2}}, q_{\mathsf{CO}}, q_{\mathsf{FN}}$によって求められる。以下に、それぞれの関係を示す。

$q_{\mathsf{FN}} = Degrad(\{q_{\mathsf{RS}}\})$
$q_{\mathsf{OV}} = Degrad(\{q_{\mathsf{RS}}\})$
$q_{\mathsf{I\ F}} = Degrad(\{q_{\mathsf{RS}}, \ q_{\mathsf{I\ F}}\})$
$q_{\mathsf{FS}} = Degrad(\{q_{\mathsf{RS}}, \ q_{\mathsf{OV}}\})$

$q_{\mathsf{F1}} = Degrade(\{q_{\mathsf{FN}},\ q_{\mathsf{FS}}\})$

$q_{\mathsf{F2}} = Degrade(\{q_{\mathsf{FN}},\ q_{\mathsf{FS}}\})$

$q_{\mathsf{CO}} = Degrade(\{q_{\mathsf{FN}},\ q_{\mathsf{IF}},\ q_{\mathsf{F1}}\})$

$q_{\mathsf{I\ NT}} = Degrade(\{q_{\mathsf{FN}},\ q_{\mathsf{CO}},\ q_{\mathsf{F2}}\})$

こうして定義した RT, AGT, PT, DEG などを用いることにより、与えられた制約を満たす実行モデルを形式的に記述することが可能となる。次章ではこれについて述べる。

第 4 章

時間制約を含む実行可能モデルの作成法

4.1 実行可能グラフの定義

実際にソフトウェア開発を行うことのできる実行計画を示す実行可能グラフを導入する。
実行可能グラフ (EG) は、PCP に時間制約と作業員制約を追加したものである。
実行可能グラフを形式的に記述したものを以下に示す。

定義 4.1

$EG = (PE, E, PT, AGT)$

但し、PEは実行可能グラフの頂点の集合を示し、PCP のフェーズ構成要素の集合。

Eは実行可能グラフの辺の集合。

PTはフェーズ構成要素に要求時間を割り当てる関数。

AGTはフェーズ構成要素に作業員を割り当てる関数。

PCP から実行可能グラフへの概念図を 4.1に示す。

4. 2 問題の定義

実際のソフトウェア開発プロジェクトでの実行計画を示すということは、以下のような条件を満たす**時間制約付き組合せ問題**であると定義する。

- フェーズ中心型プロセス上のフェーズ構成要素の依存関係に従う。

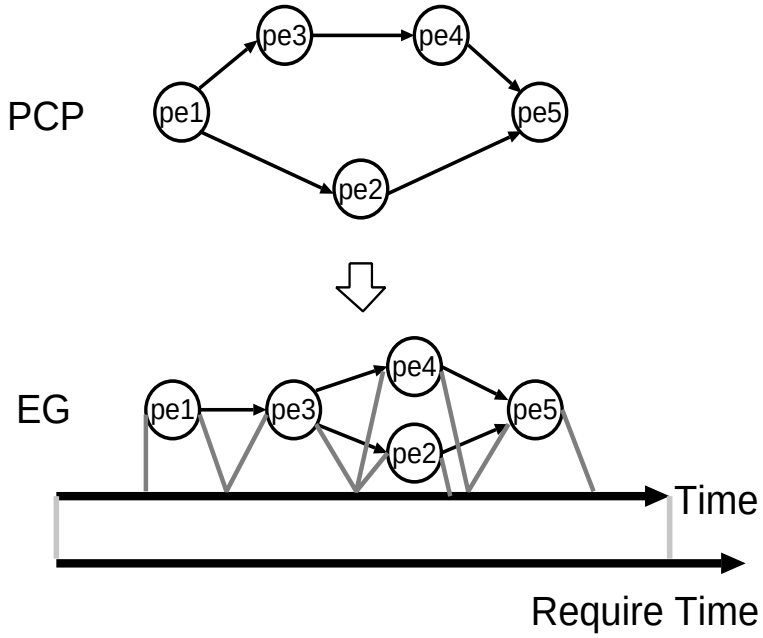

図 4.1: フェーズ中心型プロセスから実行グラフ

- 開始から終了までの各フェーズ構成要素の要求時間を合計したものがクライアントに要求される時間内におさまる。
- プロダクトの品質 (クライアントからの要求通りのプロダクトかどうか) を保つ。

4.3 実行可能グラフの作成法

この節では、実行可能グラフの作成法を記述する。記述する前に、用いる変数、関数や言葉の定義をする。

先行フェーズ構成要素とは、PCP あるいは EG のフェーズ構成要素の依存関係において、着目しているフェーズ構成要素の前に実行されるフェーズ構成要素のことである。図 4.2 において、pe5 の先行フェーズ構成要素は pe3, pe4 である。

EG で着目するフェーズ構成要素 pe の先行フェーズ構成要素 pe' を求める関数 $Prep$ は次のようになる。

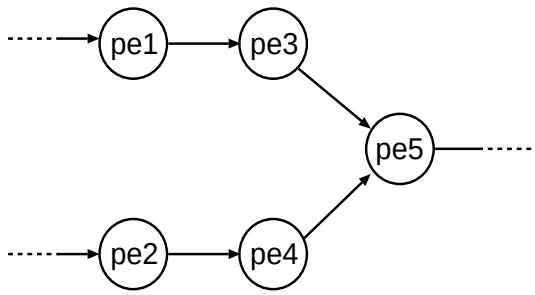

図 4.2: 先行・後続フェーズ構成要素の例

定義 4.2

$$Prep(pe) = \{pe' \mid (pe',\ pe) \in E\}$$

但し、pe は着目するフェーズ構成要素

pe'は先行フェーズ構成要素

また、**後続フェーズ構成要素**とはPCP あるいはEG のフェーズ構成要素の依存関係において、着目しているフェーズ構成要素の後に実行されるフェーズ構成要素のことである。図 4.2 において、pe1 の後続フェーズ構成要素は pe3 である。

PCP で着目するフェーズ構成要素 pe の後続フェーズ構成要素 pe''を求める関数 $Sucp$ は次のようになる。

定義 4.3

$$Sucp(pe) = \{pe'' \mid pe'' = Pe(s''), pe = Pe(s), (s,\ s'') \in \sqsubseteq\}$$

但し、pe は着目するフェーズ構成要素

pe''は後続フェーズ構成要素

EG で着目するフェーズ構成要素 pe の後続フェーズ構成要素 pe''を求める関数 $Suce$ は次のようになる。

定義 4.4

$$Suce(pe) = \{pe'' \mid (pe, pe'') \in E\}$$

但し、pe は着目するフェーズ構成要素

pe''は後続フェーズ構成要素

さらに、EG においてあるフェーズ構成要素に後続フェーズ構成要素があるかないかを調べる関数 Eqs を追加する。

定義 4.5

$$Eqs: PE \rightarrow bool$$
$$Eqs(p\,e) = \exists p\,\acute{e},\ p\,\acute{e} \in Suce(p\,e)$$

また、二つのフェーズ構成要素 $p\,e\,p\,\acute{e}$ に割り当てられた各作業員が同一人物かどうかを求める関数を Eqa とする。

定義 4.6

$$Eqa: (PE \times PE \times AGT) \rightarrow b\,o\,o\,l$$
$$Eqa(p\,e\,p\,\acute{e}) \stackrel{\triangle}{=} AGT(p\,e) = AGT(p\,\acute{e})$$
但し、$p\,e\,p\,\acute{e} \in PE$

作成される複数の実行可能グラフの中で全体の時間が最も短い実行可能グラフを**最短時間実行グラフ**と呼ぶ。

開発プロジェクトはクライアントに要求された期日までに終了しななければならないが、プロジェクトの開始時期からその要求された期日までの期間のことを**プロジェクト期間**と呼ぶことにする。

EG における各フェーズ構成要素の開始時間と終了時間を求める関数 Pst,Pet を求める。開始時間は、先行フェーズ構成要素の終了時間に等しく、終了時間はその開始時間に要求時間を加えた時間に等しい。

定義 4.7

$$Pst(p\,e) = Pet(\ Pre_p(p\,e)\ )$$
$$Pet(p\,e) = Pst(p\,e) + PT(p\,e)$$
但し、$p\,e\,p\,\acute{e} \in PE$

また、フェーズ構成要素のある集合 $X \subseteq 2^{PE}$ の要素の中で終了時間が最も遅いフェーズ構成要素を求める関数を $last$ とする。

定義 4.8

$$last : 2^{PE} \rightarrow PE$$
$$last(X) = pe \quad where \quad PT(pe) = t_0 \wedge t_0 = \max_{e \in X} PT(e)$$

フェーズ構成要素からなる集合 $Y \subseteq 2^{PE}$ から作業員別のフェーズ構成要素を求める関数を Sa とする。

定義 4.9

$$Sa : 2^{PE} \rightarrow PE \times DEV$$
$$Sa(Y) = \bigcup_{AGT(pe), pe \in Y} \{AGT(pe)\}$$

以下に実行可能グラフの求め方を示す。また、make_graph はフェーズ構成要素を実行可能グラフに追加する方法を示している。

1. PCP のすべてのフェーズ構成要素 pe を集合 P に入れる。
2. すべてのフェーズ構成要素 pe にその要求時間 $PT(pe)$ を割り当てる。
3. すべてのフェーズ構成要素 pe に作業員 $AGT(pe)$ を割り当てる。
4. 集合 P のすべてのフェーズ構成要素が実行可能グラフに割り当てられるまで 6〜10 を繰り返す。$U := \emptyset,\ M := \emptyset,\ V := \emptyset$。

   U は、実行可能グラフに追加される可能性のあるフェーズ構成要素を一時的に格納する集合。

   M は実行可能グラフに追加されるフェーズ構成要素を格納する集合。

   V は作成途中の実行可能グラフのフェーズ構成要素を格納する集合。
5. $U := U \cup pe, \qquad P := P \backslash U$

   但し、pe は PCP の先頭のフェーズ構成要素

6. 集合U中のフェーズ構成要素で

   (a) 作業員が集合U中の他のどのフェーズ構成要素の作業員とも異なるフェーズ構成要素peである場合、
   $M := M \cup pe$

   (b) 作業員が同じフェーズ構成要素がある場合

      i. 作業員が同じフェーズ構成要素の集合$U \backslash M$から作業員別のフェーズ構成要素の集合の集合を関数Saから求める。

      ii. 各作業員毎のフェーズ構成要素の集合の中から一つずつフェーズ構成要素を可能なすべての組合せになるように選択し(図4.3)、その組合せのフェーズ構成要素の集合をMに追加し、7へ。そのときの実行可能グラフが完成すると、次の組合せの集合をMに追加するということをすべての組合せに対して繰り返す。

7. make_graph(M, V)へ。

8. $U := U \backslash M$

9. $M := \emptyset$

10. $U := U \cup pe, \quad P := P \backslash U$

    但し、先行フェーズ構成要素を持たない$pe \wedge pe \in P$

11. 同一作業員が同時にフェーズ構成要素に割り当てられていないか調べる。

    (a) 割り当てられている場合、開始時間の遅いフェーズ構成要素の開始時間を開始時間の早いフェーズ構成要素の終了時間から開始するように変更し、それに以降のフェーズ構成要素の開始・終了時間を変更する。つまり、
    $Eqa(pe', pe) = true \wedge Pst(pe') < Pst(pe) \wedge Pet(pe') > Pst(pe)$ならば、
    $Pst(pe) := Pet(pe')$

    (b) 割り当てられていない場合、なにもしない。

12. 作成された実行可能グラフの中から、その総時間が最も短いものを最短時間実行グラフに選ぶ。

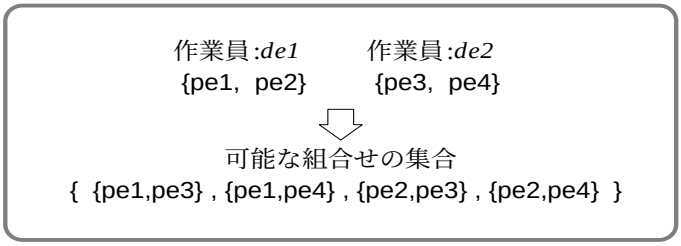

図 4.3: 可能な組合せ

13. 最短時間実行グラフの総時間とプロジェクト期間を比較する。

(a) 最短時間実行グラフの総時間の方が短い場合、そのグラフに決定する。

(b) プロジェクト期間の方が短い場合、時間短縮を図り、再度繰り返す。

make_graph(M, V)

1. $pe \in M$において

(a) $|M| = 1$ の場合、

$E := E \cup (p'e, p\phi$

ただし、$Eqs(p'e) = true \wedge p'e \in V$

$V := V \cup M$

$Ps(tp\phi), Pe(tp\phi)$を求める。

(b) $|M| \geq 2$ の場合、

i. Vの要素で EG において後続フェーズ構成要素を持たないで、peの作業員が同じ場合

$E := E \cup (p'e, p\phi$

但し、$Eqs(p'e) = fals\ e \wedge Eqa(p\ ep, 'e) = t\ rue \wedge p'e \in V$

$Ps(tp\phi), Pe(tp\phi)$を求める。

ii. Vの要素で EG において後続フェーズ構成要素を持たないで、peの作業員が異なる場合

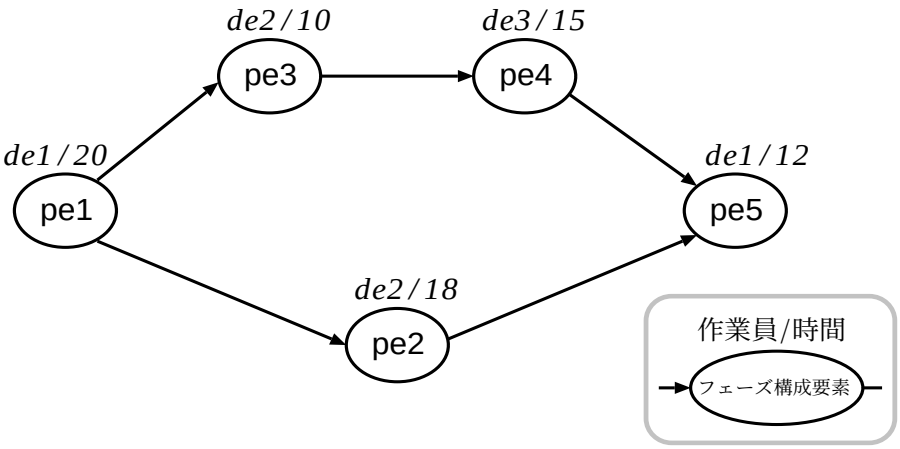

図 4.4: 実行可能グラフを求める資源付き PCP 例図

$E := E \cup (pe', pe)$

但し、$Eqs(pe') = false \wedge Eqa(pe\ pe') = false$

$\wedge\ pe' = last(X)\ \ where\ X = Pre(pe) \wedge pe' \in V$

$Ps(pe), Pe(pe)$ を求める。

$V := V \cup M$

2. Vの要素で EG において後続フェーズ構成要素を持たない要素 pe'に後続フェーズ構成要素を追加する。$pe' \in V \backslash M \wedge Eqs(pe') = false$を満たす pe'が

(a) $pe = Sucp(pe') \wedge pe \in M$を満たす場合

$E := E \cup (pe', pe)$

(b) その他の場合、 何もしない。

上記の実行可能グラフの作成法を用いて実行可能グラフの例を示す。図 4. 4は図 2. 6の PCP を基に RT, AGTを具体的に定めた図である。この PCP に要求されるプロジェクト期間を 60 日とする。この PCP 上で作業をする人は設計者 3 人 $(de1, de2, de3)$ とする。

この資源付き PCP 図から実行可能グラフを求めると 2 つのグラフが完成する。その 2 つのグラフを図 4. 5と図 4. 6に示す。このように 2 つのグラフができた理由はフェーズ構成要素 pe2,pe3 に同一の作業員 $de2$ が割り当てられているためである。そのため、フェーズ構成要素 pe2 を先に作業するか、あるいは pe3 を先に作業するかによって異なった実行

可能グラフが作成可能である。両図に示してあるように、2 つのグラフの総時間は 75 日, 60 日で図 4.6のグラフの方が総時間が短いことがわかる。そして、この PCP 図の終了期間 60 日を満たす、図 4. 6のグラフをこの PCP の実行可能グラフとして採用する。

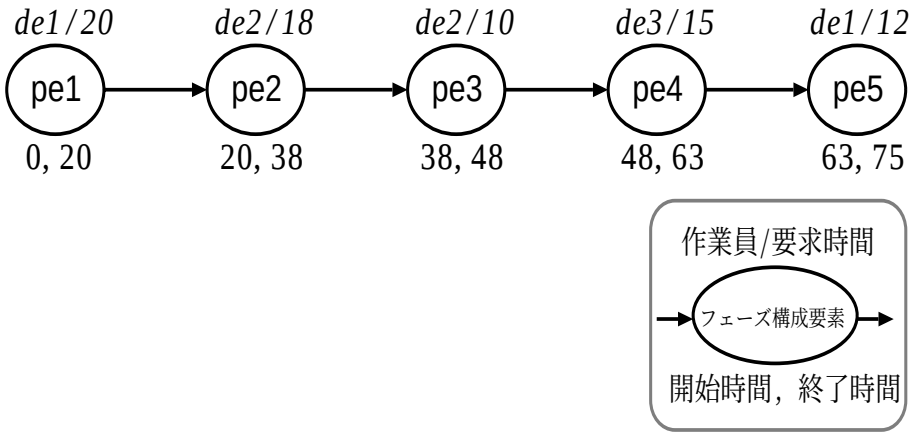

図 4.5: 実行可能グラフの候補 1

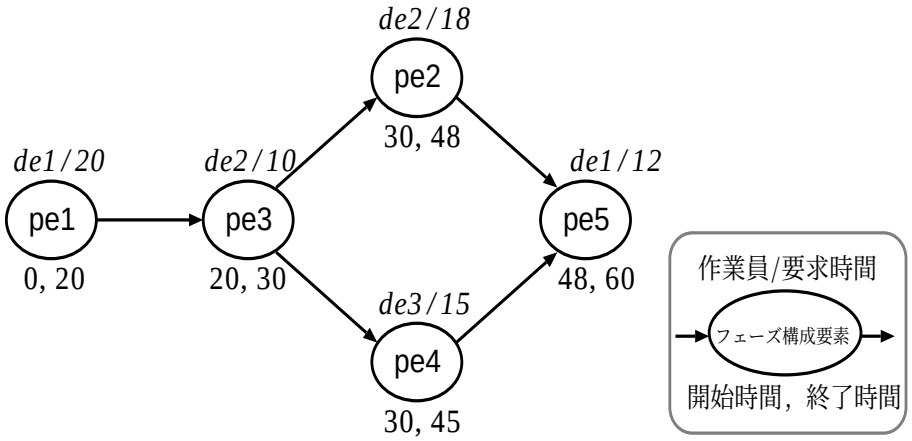

図 4. 6: 実行可能グラフの候補 2

第 5 章

EG のための時間短縮法

最初に決定した作業員の割り当てで作成された実行可能グラフがプロジェクト期間内に終了できない場合、次のような方法をとって時間短縮を図る。

5.1 作業員の再割り当てによる時間短縮

第 4 章の実行可能グラフを作成する方法では一度作業員を各フェーズ構成要素に割り当てると、作業員の割り当てを途中で変更することはない。しかし、作業員の割り当てを変更することによって、PCP で並行に実行可能な部分に同一の作業員に割り当て逐次で実行していた部分を複数の作業員に割り当てることによって、並行に実行することが可能となる。よって、作業員の割り当てを変更して実行可能グラフを作成し直すことはプロジェクト全体の時間を短縮することにつながる。

第 4 章で示した例の図 4.4のフェーズ構成要素 pe3 の作業員を $de2$ から $de1$ に変更して作成した実行可能グラフを図 5.1に示す。図 51 に示されているように、フェーズ構成要素 pe3 が pe2 と並行に実行できるようになり、全体の作業時間は 57 日と短縮したことがわかる。

しかし、すべての場合において、このように全体の作業時間が短縮できるとは限らない。逆に全体の要求時間が伸びる可能性もある。このように、作業員の割り当ては、プロジェクトの期間に大きな影響を与えることがわかる。そのため、実行可能グラフを作成するときに、PCP と作業員の制約を十分に注意しながら作業員の割り当てを行うことによって実行可能グラフの作成回数を減らすことができる。

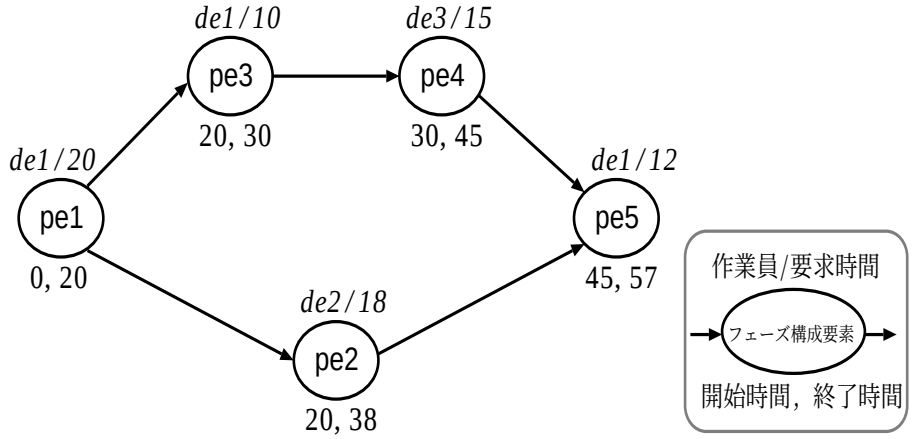

図 5.1: 作業員変更後の実行可能グラフ

5.2 フェーズ中心型プロセスの変更による方法

[6] に同じ OCP に必要とされる信頼度に応じたチェックポイントを付加することによって、PCP の構造を変更する方法が示されている。

図 3.1において、異なるチェックポイントが与えられた場合について考える。F1 をチェックポイントから除き、$CP' = \{\mathsf{FS}, \mathsf{F2}, \mathsf{CO}, \mathsf{INT}\}$ とする。この場合のステージグラフを図 5. 2に、これより得られる PCP を図 5. 3に示す。

図 5. 3は図 2. 6と比較してフェーズ構成要素の数が減っているが、フェーズ構成要素 $Pe(\mathsf{CO})$ で実行されるアクティビティ要素の数が多くなっている。一般に多くのチェックポイントを設定すればするほど各ステージでの作業量が減少し、各フェーズ構成要素の粒度が細かくなるが、フェーズ構成要素の数が増えて手戻りの発生の可能性がそれだけ多くなるというトレードオフが発生する可能性がある。

表 5. 1にフェーズ構成要素とその要求時間の対応表を示す。図 5. 3の PCP に先ほどの 3 人の作業員をそれぞれ割り当て、実行可能グラフを作成すると図 5. 4となり、全体の作業時間は 57 日になる。

このように、チェックポイントを変更することによりフェーズ中心型プロセスを変化させると実行可能グラフの全体の作業時間を短縮することができるがわかる。

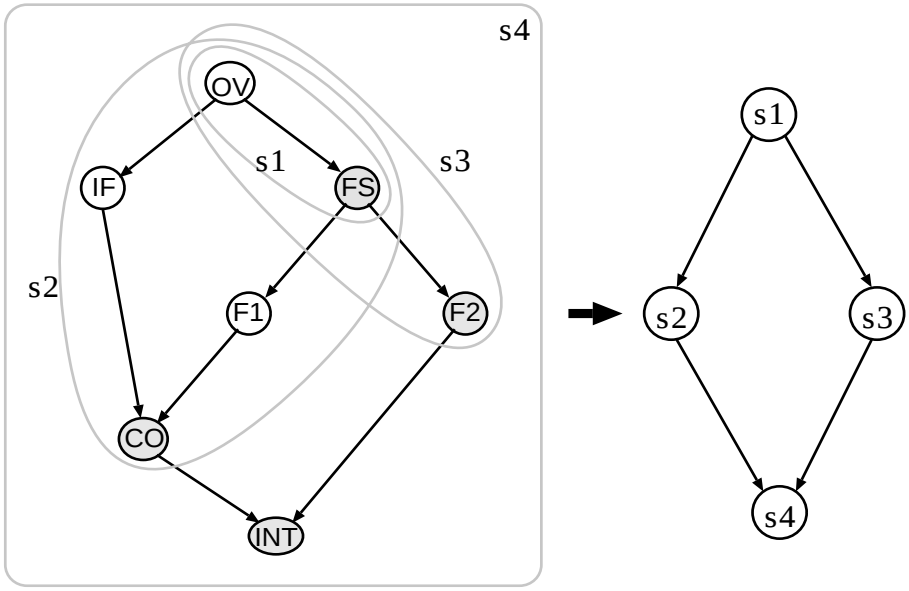

図 5.2: 適応後のステージグラフ

5.3 要求の縮小による時間短縮

ソフトウェアの発注者 (クライアント) からの要求を減らすことによって作業時間を短縮させることができる。

要求の縮小によるアクティビティ要素の要求時間の短縮は、各オブジェクト要素の劣化係数を変化させることによって、アクティビティ要素の縮小あるいは削除によって実現できる。このことから各アクティビティ要素の要求時間を短縮あるいは 0 にすることにより、フェーズ構成要素の要求時間を短縮し、プロジェクト全体の時間を短縮することができると考えられる。

ここで、時間短縮係数について述べる。各アクティビティ要素 ae の時間短縮係数を w_{ae} とする。この**時間短縮係数**は対象オブジェクト要素を生成するアクティビティ要素の要求時間の低下割合を示した実数値である。値の範囲は $0 \leq w_{ae} \leq 1$ である。対象オブジェクト要素 oe の劣化係数 $DEG(oe)$ と先行オブジェクト要素 oe' の劣化係数 $DEG(o\grave{e})$ とそれらのオブジェクト要素間のアクティビティ要素 ae からそのアクティビティ要素の時間短縮係数 w_{ae} を求める関数を CUT とすると、以下のようになる。

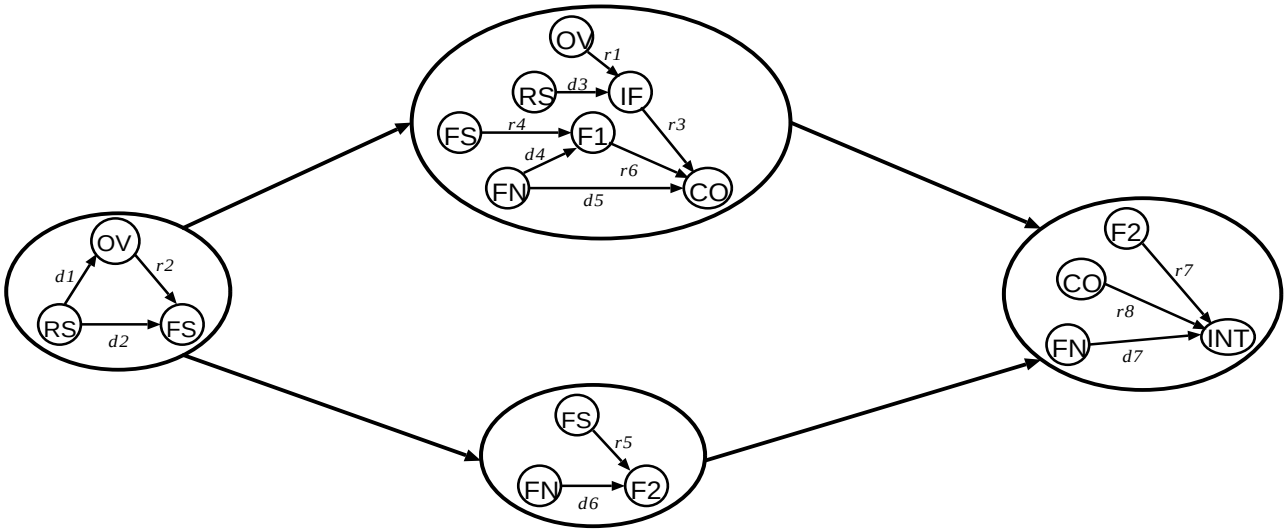

図 5.3: 適応後のフェーズ中心型プロセス

定義 5.1

$$CUT : [0,1] \times [0,1] \times AE \rightarrow [0,1]$$
$$CUT(DEG(oe),\ DEG(oe'),\ ae) = w_{ae}$$
但し、$oe = (oe,\ oe') \in AE$, $w_{ae} \in [0,1]$

ここで、図 31 の各アクティビティ要素 oe とその時間短縮係数 w_{ae}との対応表を表 5. 2 に示す。

次に、時間短縮係数を使って、要求を縮小した後の各アクティビティ要素の要求時間 $RT'(ae)$ を求める。アクティビティ要素の要求時間の縮小は、アクティビティ要素の要求時間 $RT(ae)$ とそのアクティビティ要素の時間短縮係数 w_{ae}の積をとることにより求めることができる。そのため、要求を縮小した後のアクティビティ要素の要求時間を求める関数を RT'とすると、以下のように形式的に記述できる。

定義 5. 2

$$RT' :\ AE \times CUT \rightarrow T$$

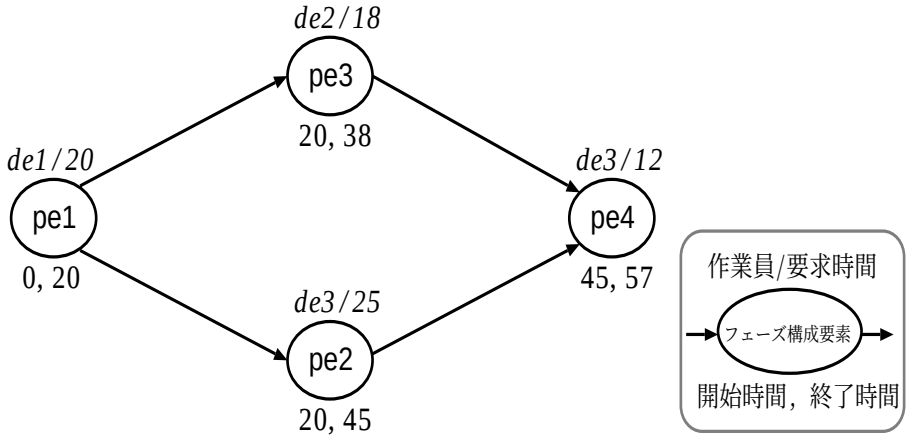

図 5.4: 適応後の実行可能グラフ

$$RT' = \{(ae, t') \mid t' = w_{ae} \cdot RT(æ),\ w_{ae} = CUT(DEG(oe),\ DEG(oe'),\ æ),$$
$$æ = (oe', oe) \in AE,\ æ,\ oe' \in OE\}$$

但し、oe は対象オブジェクト要素

oe'は先行オブジェクト要素

これにより、各アクティビティ要素の縮小した要求時間を求めることができ、定義 3.3, 3.4 よりフェーズ構成要素の要求時間を求めることができる。そのフェーズ構成要素の要求時間を PCP に付加し、再度実行可能グラフを作成する。

ここで、クリティカルパスの定義をする。**クリティカルパス**とは、プロジェクトの開始から終了までのフェーズ構成要素を連ねた経路を考えたとき、これらの所要日数の合計が最長のものをいい、その経路上にあるフェーズ構成要素をクリティカルパス要素ということにする。クリティカルパス上の時間を求める関数を CT とすると、以下のように形式的に記述できる。

定義 5.3

$$CT(pe) = \max_{pe' \in Prep(pe)} (\ CT(pe') + PT(pe)\ )$$

実行可能グラフにおける全体の時間は、要求が縮小された分だけ短縮できるとは限らない。全体の時間はクリティカルパス上のフェーズ構成要素の要求時間の総和であるので、

そのフェーズ構成要素の要求時間がどのくらい短縮されたかによるところが大きい。

ここで、要求の縮小による時間短縮の方法を示す。

1. これまで作成した実行可能グラフの中で最も全体の作業時間が短いものの PCP とそれへの作業員の割り当てを使う。
2. 要求仕様書に低下させた要求を与え、その要求に関係するオブジェクト要素の内容によって劣化係数を変更する。
3. オブジェクト要素の劣化係数とその間のアクティビティ要素の内容から時間短縮係数を見積る。
4. 時間短縮係数を用いてアクティビティ要素の要求時間を短縮する。
5. 各フェーズ構成要素の要求時間を計算する。

PE	pe1	pe2	pe3	pe4
$PT(pe)$	20	25	18	12

表 5.1: チェックポイント変更後のフェーズ構成要素とその要求時間の対応表

AE	$d1$	$d2$	$d3$	$d4$	$d5$	$d6$	$d7$	$r1$	$r2$	$r3$	$r4$	$r5$	$r6$	$r7$	$r8$
$CUT(ae)$	w_{d1}	w_{d2}	w_{d3}	w_{d4}	w_{d5}	w_{d6}	w_{d7}	w_{r1}	w_{r2}	w_{r3}	w_{r4}	w_{r5}	w_{r6}	w_{r7}	w_{r8}

表 5. 2: アクティビティ要素とその時間短縮係数の対応表

第 6 章

評価

6.1 時間制約

フェーズ中心型プロセスに対して時間制約の拡張を行った。アクティビティ要素とフェーズ構成要素のそれぞれに要求時間を記述することにより、プロジェクト全体の期間や作業をいつ始めていつまでに終わるのかといった時間などのさまざまな時間的側面を扱うことができるようになった。

6.2 要求制約

各オブジェクト要素に要求の低下を示す劣化係数を付加した。この劣化係数により、プロダクトの機能の削除や性能の低下を示すことができる。また、劣化係数はオブジェクト要素の縮小の度合を示すものでもあるので、縮小したオブジェクト要素を作成する作業時間の短縮を計算するのに使用できる。これにより、プロジェクト全体の時間も短縮することが可能となった。

6.3 実行可能グラフ

各フェーズ構成要素に要求時間を付加することにより、PCP による相対的な順序から実際に実行可能な順序を示す実行可能グラフを導くことができるようになった。実行可能グラフを求める手順は最初の作業員の割り当ての善し悪しで手順を繰り返す回数が指数的

に増加し、実行可能グラフが繰り返しの数だけ作成されてしまう。これは、PCP で並行に実行可能な部分に作業員の制約から複数の作業員を割り当てることができないときに起こる。必要とする作業員の質・量が十分であれば手順を繰り返す回数が少なくなるが、作業員には制限があるので、PCP の並行部分に十分に注意し作業員の割り当てを行う必要がある。

6.4 時間短縮法

実行可能グラフがプロジェクト期間内に全体を終えることができない場合、作業員の割り当て変更、PCP の変更、要求の低下を行うことにより全体の作業時間を短縮させプロジェクト期間内に終えることができる。

作業員の割り当て変更は、小規模のプロジェクトでは最適な作業員の割り当てを行うことが容易であるが、大規模なプロジェクトでは最適な割り当てを行うのは困難なため再割り当ての回数が増加する。そして、一回の再割り当ての計算量は多い。しかし、プロダクトの質を落とすことはない。

チェックポイントの変更によって、決められた資源量の中で条件に合うフェーズ中心型プロセスに変えることにより時間の短縮を行うことができる。これも計算量が多くなるが、プロダクトの質は落とさない。

要求の低下による方法では、劣化係数を使って各アクティビティ要素の短縮した要求時間を求め、全体の作業時間を短縮することができる。計算量も多く、プロダクトンの質も悪くなる。

第 7 章

まとめと今後の課題

7.1　まとめ

本研究では、フェーズ中心型プロセスに時間制約を形式的に示したモデルを示した。その結果、時間を考慮に入れたモデルの議論を行うことができるようになった。

また、プロダクトに対するクライアントからの要求の割合を示す要求制約についても付加した。この要求制約はプロジェクトの作業期間を短縮するのに使用した。

次に、実際のソフトウェア開発のスケジューリングに役立つ実行可能グラフをフェーズ中心型プロセスに作業員制約と時間制約を付加して作成する方法を示した。また、実行可能グラフの全体時間が、プロジェクト期間内に終了しないときの対処法として、作業員の再割り当てとチェックポイントの変更によるフェーズ中心型プロセスの変更と要求制約による 3 つの時間短縮の方法を示した。

7.2　今後の課題

- 実際のソフトウェア開発事例を用いて、このモデルの評価を行う。
- 実行可能グラフ作成のためのアルゴリズムを最適化する。
- 「コスト」を付加したときの制約付き組合せ問題の定式化を行う。

謝辞

最後に、本研究を行うにあたり終始御指導いただきました片山卓也教授には心から感謝を申し上げます。また、本研究に関して多くの有意義な御意見を頂きました鈴木正人助手ならびに片山研究室の皆様に厚く御礼申し上げます。

参考文献

[1] 片山卓也, ソフトウェアプロセスとその研究課題, 日本ソフトウェア科学会第 11 回大会論文集, pp433–436, 1994

[2] ISPW. Proceedings of the 4th International Software Process Workshop (1988)

[3] ISPW Proceedings of the 5th International Software Process Workshop (1989)

[4] Sutton, Stanley M, A Process Program in APPL/A for the Software Process Modeling Problem In Proceedings of the 6th International Software Process Workshop, 1990

[5] Katayama, T., Motizuki, S. and Yamauchi, A, Two Models for Software Process Description: Object-Centered Model and Phase-Centerd Model. In Proceedings of Information Processing society of Japan (In Japanese), 1992

[6] 鈴木正人, 片山卓也, ソフトウェアプロセス適応のための形式的モデル, コンピュータソフトウェア, Vol. 13, No.5, pp2-10, 1996

[7] 鈴木正人, 代表的なプロセス記述言語の特徴, 情報処理, vol. 36, No.5, pp392-398, 1995

[8] 望月純夫, 山内顕, 片山卓也, 人口衛生チェックアウト・システムの基本設計プロセスのプロセス・モデル HFSP による記述とその評価, 情報処理学会論文誌, vol. 33, No.5, pp691-705, 1992

[9] 竹原昭浩, ソフトウェアプロセスにおける実行フェーズ・変更方式の研究, 修士論文, 北陸先端科学技術大学院大学, 1994

[10] 柳沢滋, ER T のはなし, 日科技連出版社