社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－SE－145 (2)
2004／8／19

工程遅延発生時における対策案の自動立案 -クラッシング/ファースト・トラッキングによる対策案-

八重樫理人[†1] 木下大輔[†1] 橋浦弘明[†1]
上之薗和弘[†1] 林雄一郎[†1] 古宮誠一[†1]

ソフトウェアプロジェクトの多くはプロジェクト進行中に現在 Agile（アジャイル）と呼ばれる超短納期ソフトウェア開発の為の手法がいくつか提案がされている。これまでは先行作業が完全に終了しないうちに後続作業を開始することは、リスク回避の視点からタブーとされてきた。しかしながら超短納期でのソフトウェア開発を可能とするために、このタブーを冒してでも開発期間の短縮の為に工程遅延を回復する必要がでてきた。このため本論文ではソフトウェア開発プロジェクトにおいて工程遅延が発生した時、対策案としてファースト・トラッキング（直前の作業が終了しないうちに後続作業を開始する事）による最適なスケジュール案を自動立案する仕組みをシステムの実行例を示す事で本支援システムの有効性を明らかにしている。

Automatically Creating a Fast –Tracking-Based Countermeasure Plan against Process Delay

Rihito Yaegashi, [†1] Daisuke KINOSHITA, [†1] Hiroaki HASHIURA, [†1]
Kazuhiro UENOSONO [†1] Yuichirou HAYASHI, [†1] and Seiichi KOMIYA [†1]

Some techniques for the software development in a very short period are proposed. Former it is a taboo that Pre-work is started before post-work is completed completely in order to avoid a risk. However it is necessary to recover process delay even if it performs a taboo in order to make software development in a short period possible. This paper proposes a method to create automatically such a successful schedule plan that a project, which was behind schedule, will be completed on schedule by means of 'Fast-Tracking'. And the paper proves that the method is effective in software project management through an example of system.

1. はじめに

大規模なソフトウェアの開発は労働力を結集するためにプロジェクトを組んで行われるのが一般的である。どのようなライフサイクルモデルを採用しようとも、ソフトウェア開発プロジェクトには、開発計画（＝開発のための作業スケジュールや各作業への要員割当などに関する計画）というものが必ず存在する。従って、プロジェクトを成功へと導くためには、ソフトウェア開発計画をもとに管理目標を設定し、その達成度をフォローするという方法が効果的である。しかし、ソフトウェア開発計画通りに開発を行うことは、実は容易なことではない。何故なら，ソフトウェア開発作業は頭の中で行われるので、検討の深さが他人からは見えないからである。このため，機械化できる部分はできるだけ機械化し，プロジェクト管理者が、担当者と対話する事によって検討の深さを確認するための時間を確保する事が必要である。またソフトウェア開発においては、作業量の見積もりやプロジェクトにおけるリスクの予測等が困難であるため、ソフトウェア開発計画の立案そのものも困難であるといえる。したがって、機械化によってプロジェクト管理者の負担を軽減すると共に、ソフトウェア開発計画の立案を支援するソフトウェア開発計画自動立案システムを研究開発する。

ソフトウェア開発プロジェクトにおける工程遅延の回復方法としてこれまで使われている手法にクラッシング、縮退と呼ばれる２つの方法がある。（表１参照）

表１ 所要期間の短縮方法

Crashing クラッシング	余剰人員やコストをプロジェクトに追加投入する事で所要期間の短縮を図る方法
縮退	顧客の求めるシステムの機能の一部分を削除する事で所要期間の短縮を図る方法
Fast-Tracking ファースト・トラッキング	先行作業が完全に終了しないうちに後続作業を開始する事で所要期間の短縮を図る方法

クラッシング(Crashing)とは余剰人員や余剰コストをプロジェクトに追加投入する事によって、プロジェクト全体の所要期間の短縮を図る方法である。また縮退と呼ばれる方法は顧客の要求するシステムの機能を一部分削減することでプロジェクト全体の仕事量減らし、所用期間短縮を図る方法である。しかしながら現在のような多種多様化する顧客の要求を満足するために、さらに超短納

†1 芝浦工業大学大学院
Graduate school of Shibaura Institute of Technology

期でのソフトウェア開発(Agile)を目指す必要がある。これまでは、先行作業が完全に終了しないうちに後続作業を開始することはリスク回避の視点からタブーとされてきた。しかしながら超短納期ソフトウェア開発(Agile)ではこのタブーを冒してでも工程遅延の回復を図る必要がでてきた。

そこで本論文では工程遅延が発生した時、対策案としてファースト・トラッキング(Fast Tracking；直前の作業が終了しないうちに後続作業を開始する事）を行う事によって工程遅延が回復できるような再計画案を自動生成する仕組みを、システムの実行例を示すことによって枠組みの有効性を証明する。

本論文の構成を以下に示す。2.では、ソフトウェア開発計画立案問題がもつ制約および、制約の表現内容について述べる。そして、ソフトウェア開発計画立案問題とは上記の制約条件を満足するリソースの最適割り当て問題を解く事であるということを示す。また上記の制約条件を用いてファースト・トラッキングを実現する為の手法について述べる。3.では工程遅延を回復するためにファースト・トラッキング実施の為に必要である各工程の持つリスク、及びネットワーク上の位置によるリスク（以下ネットワークリスクと呼ぶ）の計算方法及びファースト・トラッキングを行う工程の決定方法について述べる。更に 4.では工程遅延が発生した際にファースト・トラッキングを行う事によって工程遅延が回復できるような計画案を自動生成するシステムの実行例を示すことによって，提案したシステムの枠組みの有効性を示す。5.では本論文のまとめを述べる。

2. ソフトウェア開発計画立案問題における制約とファースト・トラッキング実施の為の追加制約

2.1 ソフトウェア開発計画立案問題に置ける制約とその表現方法

ところで，ソフトウェア開発作業を行うのは人間である。しかしながら人間なら誰でもいいというわけでなく、その作業を行うことができるだけのスキル等を備えていなければ務めることができない。しかも数多くの大規模プロジェクトが存在し、且つ並行にプロジェクトが進行するような現在の状況においては、そのような有能な人には限りがあり、有能な人は多くのプロジェクトから引っ張り凧となる。このことは高度な専門知識を持った人材を数多く確保する事によってプロジェクトを組むことが初めて可能になる，ソフトウェア開発のような知識集約型の産業にして初めて出現した問題であるといえる。このため、ソフトウェア開発では、各作業を実施する上で必要となる要員（＝人的リソース）の割当条件や割当条件を満足する人的リソースの割当可能期間（＝その人のスケジュールの空いている期間）を考慮に入れてスケジューリングしなくてはならない。さらにプロジェクトに必要である計算機のような非人的リソースも同様に考慮に入れる必要がある。

従って、次の事が言える。

ソフトウェア開発の作業スケジュールは，ソフトウェア開発のための各作業を実施する上で必要となる人的及び非人的リソースの割当条件（これをリソースの割当条件に関する制約と呼ぶ）や割当条件を満足する人的及び非人的リソースの割当可能期間（これをリソースの割当可能期間に関する制約と呼ぶ）に依存する。

そこで以下ではソフトウェア開発計画立案の際に生じる問題を制約と捉え、ソフトウェア開発計画立案問題の持つ制約の内容を具体的に記す。

(1)作業順序に関する制約

ソフトウェア開発の各工程には、中間成果物を媒介して、それらの実施順序が決定する。 例えば、図 1 の工程 b を例に挙げて説明する。工程 b を実施するには、工程 a による成果物(中間成果物)αが工程 b に着手する前に出来ていなければならない。これを工程 b の事前条件と呼ぶ。また、工程 b の成果物(中間成果物)βが工程 c に着手する前に生成されていなければならない。これを工程 b の事後条件と呼ぶ。

中間成果物α、βによって工程 a,b,c の実施順序が決まる。このような制約を作業順序に関する制約と呼ぶ。

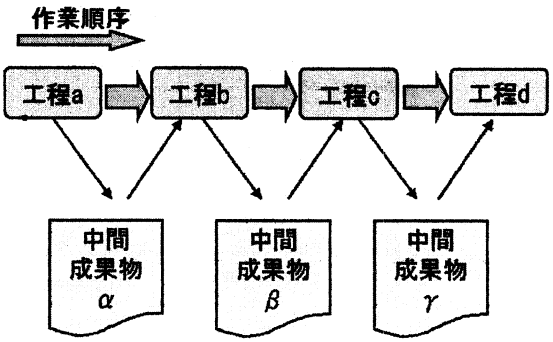


図 1.中間成果物による工程の順序制約

(2)リソースの割り当て条件に関する制約

ソフトウェア開発の各作業には、その作業を実施する上で必要をなるスキル、資格、機能などをもつ人的リソース（要員）または非人的リソース(マシン環境など)しか割り当てることができない、という制約(＝リソースの割当条件に関する制約)がある。例えばプログラム開発言語や、システムのテスト、デバッグ作業などの工程はそれぞれ能力を持つものが担当する必要がある。このため、ソフトウェア開発の作業スケジュールは、ソフトウェア開発のための各作業がもつ人的リソースと非人的リソースの割当条件に関する制約に依存する。

ところで「ある客先で大きなトラブルがあった部下を同じ客先に連れて行くことを避ける」という制約は，(2) の Constraints concerning the conditions for resource allocation に分類される。これは何故なら，ある客先で大きなトラブルがあった部下というのは，広い意味で，そのリソースに割り当てられるだけの資格がないという事ことになるからである。

(3) リソースの割当可能期間に関する制約

ソフトウェア開発の各作業には、割当条件を満足するリソースでも、そのリソースにとって割当可能な期間(＝空きスケジュール)にしか、そのリソースを割り当てることができない、という制約(＝リソースの割当可能期間に関する制約)がある。リソースの割当可能期間に関する制約については文献[1]で述べられている。

従って A1,A2,A3 のすべての作業が正常終了する確率 A は $A=(1-\alpha)(1-\beta)(1-\gamma)$で表される。直列構成の場合の作業 X の着手に係るリスク R は $R=1-A=1-(1-\alpha)(1-\beta)(1-\gamma)$となる。

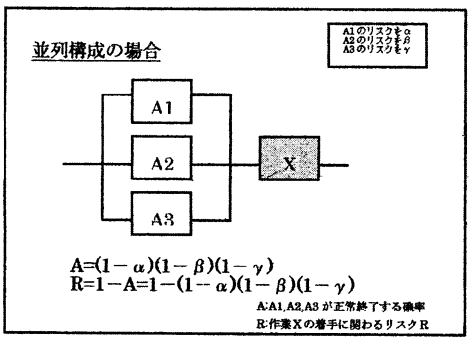


図 3. 並列構成のリスクの計算方法

図 3 のような並列構成でも作業 X が着手可能であるためには、X の先行作業 A1,A2,A3 がすべて正常に完了していなければならない。したがって A1,A2,A3 のすべての作業が正常終了する確率 A は
$A=(1-\alpha)(1-\beta)(1-\gamma)$で表される。

従って並列構成の場合の作業 X の着手に係るリスク R は $R=1-A=1-(1-\alpha)(1-\beta)(1-\gamma)$となる。よってネットワークリスクの計算方法は直列構成でも並列構成でも同じであり図 4 のようになる。

ネットワークリスク

$$R_i = 1 - \prod_{j=1}^{i} A_j$$
$$= 1 - \prod_{j=1}^{i} (1 - r_j)$$

r_j:スケジュール表(ネットワーク)上の各ノード(工程)のリスク

A_i:スケジュール表(ネットワーク)上の各ノード(工程)の稼動率

R_i:スケジュール表(ネットワーク)上の位置による各ノードのリスク

図 4. ネットワークリスク

3.3 ファースト・トラッキングを行う工程の決定方法

3.2 で求められたネットワークリスクを基にファースト・トラッキングを行う工程が決定される。クリティカルパス上の作業で、予め管理者により定義されたネットワークリスク値より低いリスクを持つ工程に対してファースト・トッラキングを行うことで、工程遅延の回復案の立案を図る。この際予め定義されるネットワークリスク値は、プロジェクトマネージャーによって与えられる事とした。またファースト・トラッキングを行う工程を担当する要員は、先行作業を担当する要員とスキルが同等かそれ以上であることを条件とした。これはファースト・トラッキングを行う際、担当する要員は先行作業についても十分な知識が必要である為である。

4. ファースト・トラッキングを行う工程の遅延回復計画の自動立案とその評価

本章では,工程遅延発生した時に、直前の作業が終了しないうちに後続作業を開始する事(ファースト・トラッキング)を行うことで、工程遅延を回復させる計画案の自動生成例を示すことにより、本システムの枠組みがソフトウェア開発計画立案問題の解決に有効であることを示す。

[作業概要]
例題プロジェクトとして、8 月 1 日 から開催されるある展示会用のデモシステムを開発するプロジェクトを考える。このプロジェクトは 7 月 2 日に組織され、7 月 31 日までにシステムを完成させなくてはならない。システム開発のために行わなくてはならない作業は、基本設計、概要設計、概要設計レビュー、詳細設計、コーディング、単体テスト、統合テストである。プロジェクト計画立案の際に考慮すべき制約を以下に列挙する。デモシステムは高性能を要求されたシステムであるため、詳細設計の一部分に対して性能見積もりを行いコーディング作業に反映させる。性能見積もりは性能解析技術を必要とする。展示会で使用するハードウェアは今回出展のための開発マシンであり 7 月 26 日にならないと利用できない。しかもこのマシンは 7 月 31 日には展示会場に搬出されなくてはならない。展示会で使用するマシンの使用期間に関する制約によりテストを効率よく行うためにテスト支援ツールの開発を行うことになった。このような状況下の中で計画の立案を行い、プロジェクトを開始した。例題プロジェクトの人的リソースはそれぞれ以下の制約をもつ。

・プロジェクト管理者(Project Manager)はプロジェクトの管理、進捗の管理、承認活動をおこなう。
・設計技術者(Design Engineer)の基本的な責務は設計とコーディング作業である。
・品質保証技術者(Quality Assurance Engineer)の基本的な責務はテスト計画とその実施である。

表 3 プロジェクトで使用する人的リソースの属性値

	SKILL		Available time	Cost
A	System analyst Design engineer Programmer	a b c	-7/18	2800
B	System analyst Design engineer Programmer	b a a	Any	3500
C	System analyst Design engineer Performance analyst	a b c	7/10-7/18 8/1-8/5	3000
D	Design engineer Programmer	b a	7/2-7/20 7/25	3000
E	QA engineer	b	Any	3200
F	QA engineer	a	Any	3600
G	System analyst Design engineer Programmer	a a b	7/10	4000
H	System analyst Design engineer Programmer	b a c	7/17	3300

表 4 プロジェクトで使用する非人的リソースの属性値

Name	Available time
Performance Analysis Tool	7/10-7/18.8/1-8/5
Test Support Tool	-7/23
Machine for Development	Any
Machine for Exhibition	7/24-7/30

各役割を担う要員として各設計技術者には A、B、C、D、G、H の各 6 名、品質保証技術者には E、F の 2 名が割り当てられ、その役割に応じた作業を実行する。またプロジェクトでは開発マシン、展示会で稼動するマシン、性能見積もりを行う際に使用するツール、テストを効率的に行うために開発するツールを使用する。リソースに関しては表 3 と表 4 に示したような制約がある。

(4) リソースの能力的限界に関する制約

各リソースの能力的限界を表現するために" capacity"という概念を導入し、リソースの属性として表現する。具体的には、容量をそのリソースの稼動率(単位は%)の上限値で表現する。すなわち、そのリソースが一日に割り当てられた作業時間の合計(同一のリソースが並列に進められる複数の作業に同時に割り当てられた場合にそれらの合計)を、そのリソースが1日に稼動可能な時間で割り、その値に100を掛けることによって得られる値を稼働率とする。そして、あらかじめ設定されたこのリソースにおける稼働率の上限をもって、このリソースの例えばある作業者P1に対して、1週間(5日)に作業A(所要日数2日)と作業B(所要日数2日)の2つが割当てられていたとすると、作業者P1のその週の稼働率は80%となる。このときこのリソースの上限稼働率が80%以上であればこれらの作業は割当て可能であるが、80%より小さいと割当て不可能となる。このようにすれば、稼働率を作業者の負荷状況を示す尺度として利用することができ、作業者に対する過度の作業の割当てをチェックできる。非人的リソースに対しても同様の概念を導入する。なお容量(上限稼働率)は一般にリソースのランクによって異なると考えられる。

それぞれの制約をすべて満たすような開発計画を立案することは大変難しい。従って、ソフトウェア開発計画立案問題とは上記の制約条件を満足するリソースの最適割り当て問題を解くことだといえる。

2.2 ファースト・トラッキング実施の為の追加制約

2.1でソフトウェア開発計画立案問題は、制約を満足するリソースの割り当て問題であるという事を述べた。我々が現在開発中であるソフトウェア開発計画自動立案システムにおいて、ファースト・トラッキングによる工程遅延の回復案立案の為には2.1(1)で述べた作業順序に関する制約条件を見直す必要がでてきた。従来、我々のシステムにおいては後続作業となる工程は先行作業が完了する事が作業開始の条件であった。そこで後続作業は先行作業の完了を待たずに、後続作業が開始できるように制約条件を改めた。このようにする事で後続作業は先行作業の完了を待つことなく、作業が開始できるような開発計画が立案されるようになった。

工程遅延の回復案をファースト・トラッキングで生成する場合、先行着手される工程実施者に関して新たにリソースの割り当て条件に関する制約加えた。工程作業を先行着手する場合、中間成果物がきちんと予定通りに得られる保証が得られない。したがって先行着手される工程を担当する要員は前の工程に関する十分な知識が必要となる。さらに付け加えると、先行作業に対して後続作業を担当する要員が最低限守るべき条件を加えなくてはならない。そこで先行着手される工程を担当する要員は、直前の工程を実施できるスキルを有し、かつスキルレベルにおいても前の工程を担当する要員と同じかそれ以上のレベルを有するという制約条件を新たにつけ加えた。

3. 各工程作業のファースト・トラッキング（早期着工）の実現方法

我々が研究開発中のソフトウェア開発計画自動立案システムでは、2章で述べたようにこれまではソフトウェア開発の各工程には中間成果物を媒介して、それらの実施順序が決定するという制約条件に基づいて開発計画の立案を行ってきた。しかしこれまでは、先行作業が完全に終了しないうちに後続作業を開始することはリスク回避の視点からタブーとされてきた。しかし最近のAgile(アジャイル)と呼ばれる超短納期ソフトウェア開発ではこのタブーを冒してでも工程遅延の回復を図る必要が出てきた。そこで現在開発中のシステムにファースト・トラッキング（直前の作業が終了しないうちに後続作業を開始する事）を行える追加機能を付加する事で、今まででは実現することができないような短い開発期間で、ソフトウェア開発可能な開発計画案が立案される仕組みについて以下に述べる。

3.1 各工程のリスクの見積もり

本研究では各工程のリスクを0から1までの数値で見積もり、1は失敗の確率が最も高い場合であるとした。各工程におけるリスクは、1に近ければ近いほどその工程は失敗する可能性が高い事を意味し、また0に近ければ近いほど失敗する可能性が低い事を意味している。各工程のリスクはあらかじ表2のような形で見積もられている。たとえば表2においては、PEの見積もられたリスクが0.08であるので全工程の中で最も高く失敗の可能性が一番高い工程である事を意味し、またSD1の見積もられたリスクが0.008であるので全工程の中で最も失敗の可能性の低い工程である事を示している。本システムでは各工程が持つリスクは事前に見積もられ、システムに与えられている。各工程のリスクの見積もり方法は文献[3]のとおりである。

表2. 見積もられたリスクの例

工程	見積もられたリスク
SA	0.01
SD1	0.008
SD2	0.03
DR	0.01
DD1	0.01
CT1	0.02
DD2	0.02
PE	0.08
CT2	0.03
DT	0.04
IT	0.02

3.2 ネットワークリスクの計算の考え方

先行作業が完全に終了した後に後続作業を行う場合に比べて、ファースト・トラッキング（先行の作業が終了しないうちに後続作業を開始する事）を行った場合は、先行作業が予め定められた仕様通りに中間成果物が得られる保証がないのでリスク（失敗可能性の予測値）が当然高くなる。それでもファースト・トラッキングを導入せざるを得ない場合には、工程作業の中でネットワーク上の位置によるリスク（以下ではネットワークリスクと呼ぶ）が少ない工程を対象にファースト・トラッキングするとよい。ネットワークリスクの計算方法について以下に述べる。

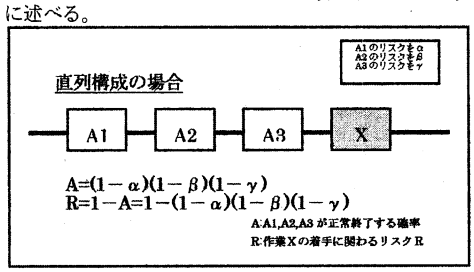


図2. 直列構成のリスクの計算方法

図2の作業Xが実施される為のリスクについて考える。A1,A2,A3各工程の持つリスクはあらかじめα,β,γと与えられている。直列構成では作業Xが実施されるためにはA1,A2,A3すべて正常に完了していなければならない。

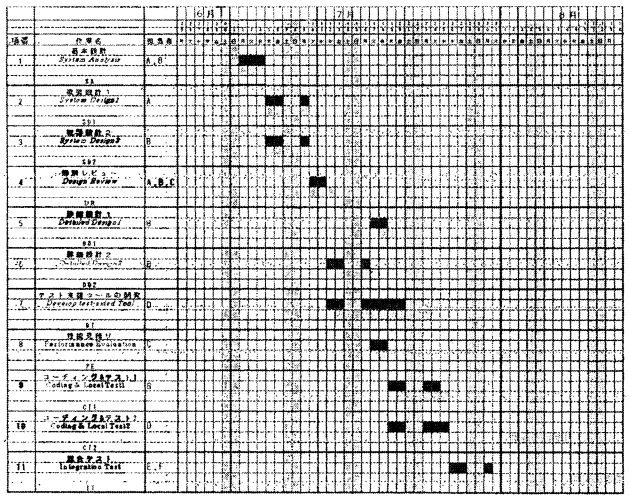
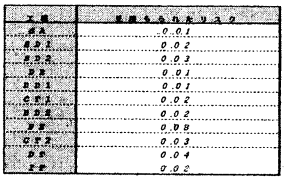


図5 初期の開発計画と遅延発生後の開発計画

表3 本例題における見積もられたリスク



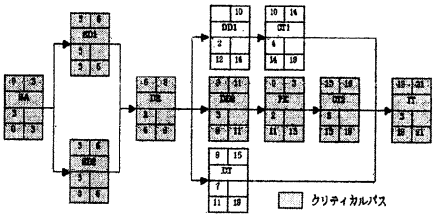


図6 PDM 法によるクリティカルパスの計測

本研究では表 3 のように作業者の能力を 4 段階(よくできる a、できる b、不得手 c、遂行不可能 d)にて評価することとした。ここでの”よくできる”能力とは一人で素早く作業する事ができ、かつ不得意な人を指導しながら作業する事もできる人、“できる”は他人を指導する事はできないが、一人で作業することはできる人を意味する。“不得手”は他人の指導がなければ作業する事ができない人を意味する。“遂行不可能”とはプロジェクトの遂行中に他人の指導があっても、作業の遂行が不可能な人を意味している。

今回は図 5 のような開発計画で行われているプロジェクトにおいて開発計画の遅延が起きているとする。図 5 の計画に基づいて開発が進められていたが、7/11 の終了時点で開発計画に遅延が生じた。クリティカルパス上での工程遅延は、開発計画全体での遅れを意味している。ところで DR(Design Review)は、クリティカルパス上作業であるので、DR での工程遅延は開発計画全体での遅れを意味する (図 6 参照) 。図 5 をみれば、1 日の工程遅延が、結果的に 15 日間の遅延となることがわかる。表 3 を見ればわかるように、これは PE(Performance Evaluation)の作業を担当する要員 C の割り当て期間に関する制約により、要員 C が(Performance Analysis Tool も) 8/1 まで PE の作業を行うことができないからである。
そこで 7/12 以降の工程に対してファースト・トラッキングを行うことで工程遅延を回復するような対策案を立案する。PDM 法で表記された図 6 をみればわかるように、クリティカルパス上の作業は DD2,PE,CT2,IT であることがわかる。マシンの使用期間(7/26-7/30)の制約から IT は初期の計画通りに作業を行わなくてはならないことがわかる。しかしながら IT(Integration Test)は最終的な統合テストであり、特別の技術を必要とする作業であるので、特別な技術を持つ要員を割り当てなくてはならない。ファースト・トラッキングできない制約条件を有している。そこで DD2,PE,CT2 のいずれかの作業に対してファースト・トラッキングを実施し、7/31 までに紀開発が終了する計画案を生成する。この時ファースト・トラッキング実施可能なネットワークリスクは 0.1 以下とした。

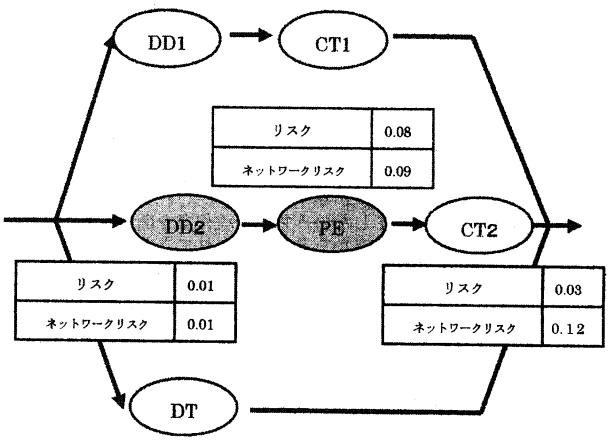


図7 ファースト・トラッキング実施工程の決定方法

本例題プロジェクトについて考える。本例題では 2 つの工程遅延を回復する再計画案が 2 つ生成された。ここで採用された再計画案図 8 をみると工程遅延が回復できている事がわかる。図 8 の再計画案 1 は DD2 をファースト・トラッキングしていることがわかる。また DD2 は初期計画では B が担当する作業であったが、G が担当している。G は DR の作業中に DD2 の作業を初期計画通りに行っている。このとき図 8 の再計画案 1 がファースト・トラッキングを行う際の制約条件をみたしているのかを考える。本支援システムではネットワークリスク 0.09 以下の工程作業にたいしてファースト・トラッキングを行う事を許している。図 7 を見ると DD2 のネットワークリスクは 0.01 でありこの制約条件を満足している。ファースト・トラッキングを行う要員のスキルにおいても、担当である要員 G は先行作業 DR のスキルにおいても先行作業を担当する要員 A,B と同等以上であるので制約条件を満足している。

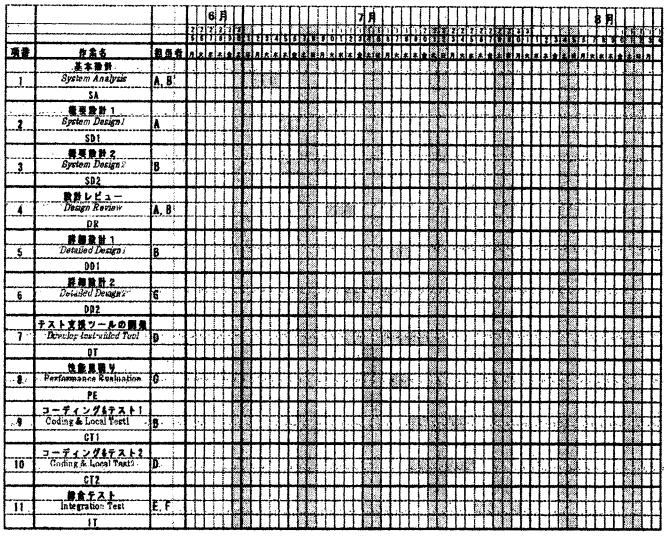


図8 採用された開発計画(再計画案 1)

再計画案 2 は PE に対してファースト・トラッキングを行う計画案が立案された。図 7 をみると PE のネットワークリスクが 0.09 でありファースト・トラッキングが実施可能である事がわかる。この時 PE は、初期計画通り要員Cが担当している。要員 C は DD2 のスキルに関しても要員Bと同等以上であるので、実施可能である。

今回の再計画案では CT2 にファースト・トラッキングを行う計画案が生成されなかった。これは図をみるとネットワークリスクが 0.12 であり、0.09 以下という制約条件を満足していないからである。

今回の例題では再計画案 1 が採用された。これは

[コストが最低となるようにする。] を再計画案立案のための戦略としたからであるが、これは GA の評価関数を変更する事によって他の戦略を設定できる事を意味している。

5. おわりに

本論文では、ソフトウェア開発計画立案問題を解決するために、ソフトウェア開発計画立案を行う際に考慮すべき制約を明らかにすると共に、制約の表現内容について述べた。また工程遅延が発生したときにこれまでは、先行作業が完全に終了しないうちに後続作業を開始することはリスク回避の視点からタブーとされてきた。しかし最近の Agile(アジャイル)と呼ばれる超短納期ソフトウェア開発ではこのタブーを冒してでも工程遅延の回復を図る必要が出てきた。そこで工程遅延発生時の対策案の立案のために、ファースト・トラッキング（直前の作業が終了しないうちに後続作業を開始する事）によって工程遅延が回復できるような計画案を自動生成する仕組みをシステムの実行例を示すことによって、提案したシステムの枠組みの有効性を示した。

謝辞
この研究に、日頃協力して戴いている芝浦工業大学情報工学科知能ソフトウェア研究室の関係各位に感謝します

参 考 文 献

1) 古宮 誠一,澤部 直太, 櫨山 淳雄"制約に基づくソフトウェア開発計画の立案", 電子情報通信学会論文誌 D-I Vol.J79-D-I No.9, pp.544-557, 1996.
2) S. Komiya A. Hazeyama, "A Meta-Model of Work Structure of Software Project and a Framework for Software Project Management System, "IEICE Tran INF SYST ,vol.E81-D.No12,pp1415-1428,Dec 1998
3) W. H.ローツハイム（著）深沢十郎（訳）Structured Computer Project Management(原書)構造化プロジェクト管理, 近代科学社, 1992
4) 八重樫理人, 原田 将来, 野田 寛之, 齋藤 匠, 古谷慶太, 古宮 誠一, "GA によるソフトウェア開発計画自動立案", 信学技報 TECHNICAL REPORT OF IEICE KBSE2002-48(2003-3), pp.27-32,2003
5) 八重樫 理人, 木下 大輔, 橘浦 弘明, 上之薗 和宏, 古宮 誠一, "工程遅延発生時におけるファースト・トラッキングによる対策案の自動立案" 信学技報 TECHNICAL REPORT OF IEICE KBSE2003-42(2004-1), pp.7-12,2004
6) 八重樫 理人, 木下 大輔, 橘浦 弘明, 上之薗 和宏, 古宮 誠一, "制約に基づくソフトウェア開発計画自動立案システム-工程遅延発生時における対策案の立案-" 信学技報 TECHNICAL REPORT OF IEICE KBSE2003-18(2003-18), pp.29-34,2003
7) Liu,L and Horowitz, E., Object Database Support for a software Project Management Environment, ACM SIGSOFT Software Engineering Notes Vol.13, No5, pp.85-96(Nov.1988)A. Gupta, R. Jain: Visual Information Retrieval, *Communications of the ACM*, Vol.40, No.5, pp.71-79 (1997).
8) Matsumoto Y and Ajisaka T, A Data Model in the Software Project Database KyotoDB, JSSST Advances in Software Engineering Environments, (the International Conference on Software Engineering pp.150-157 (1985)
9) Penedo M.H.and Stuckle, E.D., PMDB - A Project Master Database For Software Engineering Environments, 8th International Conference on Software Engineering, pp.150-157 (1985).