Vol.2013-SE-179 No.24
2013/3/12

同一期間に複数のプロジェクトに従事する作業者の作業優先順序決定支援

比護徹治[†1]　森一樹[†1]　橋浦弘明[†2]　古宮誠一[†1]

企業では少ない作業要員でより多くのソフトウェアを開発するために，一人の作業者が同時期に複数のプロジェクトに携わることが少なくない．一人の作業者が同時期に担当する複数の作業がそれぞれ別のプロジェクトに属する場合，この作業者は，自身が所属するそれぞれ別のプロジェクトのマネージャの作業指示に従わなければならない．そのため，状況によってどのマネージャの作業指示に従うべきかを自分で判断しなければならず，この作業者の判断能力とスケジュール管理能力が問われる．そこで，このような状況にある作業者が，それぞれ別のプロジェクトに属する複数の作業に対して，どのような優先順序を付け，どのようなスケジュールで実施すべきかの決定を支援する方法を提案する．

A Decision Support System for Prioritizing Tasks of a Worker Who Works on Several Projects Simultaneously

Tetsuharu Higo[†1]　Kazuki Mori[†1]　Hiroaki Hasiura[†2]
And　Seiichi Komiya[†1]

1. はじめに

現在，ソフトウェア開発の現場では，大規模で複雑なソフトウェアを短納期で開発する要請が多く，それを実現するために複数のプロジェクトの並行開発が一般的に行われている，短納期での並行開発は人員不足を引き起こす原因となり，この対策として他のプロジェクトの作業に従事する人員が，人員不足の生じたプロジェクトに同時に関わるという状況が珍しくない．そのような状況において，作業の競合（割り当てられた作業のスケジュールが日程的に重なってしまうこと）が発生するが場合ある．このとき作業者はどちらの作業を優先すべきかの判断を迫られるが，場合によっては上司の承認が必要なものもあり，その判断は難しい．そのため本研究では，工程の持つ属性に着目し，後続工程への影響度を算出することにより全体最適なスケジュール案を提案する．それにより作業の優先順序の決定を支援する手法を提案し提案手法の検証を行う．

以下に，本稿における次章以降の構成を示す．２章ではソフトウェア開発計画が持つ制約について述べる．３章では後続工程に及ぼす影響の大きさから見た工程の分類と定義を述べる．４章では本研究の契機となった事例についてのべる．5 章では一人の作業者が担当する複数の工程が競合したときのスケジュールの考え方について提案する．６章では支援システムの利用イメージを示す．７章ではまとめと今後の課題について述べる．

†1 芝浦工業大学大学院
Graduate School of Science and Engineering Shibaura Institute of Technology
†2 東洋大学
Toyo University

1.1 本研究で前提とするプロジェクトの状況

本研究で前提とするプロジェクトの状況を以下に述べる．ある作業者が同一期間に２つのプロジェクトＸとＹに従事しているとき，以下の 2 つの状況が考えられる．

(1)　X，Y 両方を管理する同一のマネージャが存在するこのようなマネージャは作業者のスケジュールを把握できているため，作業の競合が発生したときには，作業者はマネージャの指示通り作業を遂行すればよい．

(2)　X, Y 両方を管理する同一のマネージャが存在しないこの場合，X を管理するマネージャはプロジェクト Y の詳細を把握できておらず，両方のプロジェクトを把握しているのは作業者自身だけである．そのため，プロジェクト X，Y，の作業が競合したとき，作業者は自分の判断で行動しなければならない．このとき，作業者は両方のプロジェクトの作業の納期を守るため，作業のスケジュール案をそれぞれのマネージャに提示して，承認を仰ぐ必要がある．本研究ではこのような状況を前提に議論を進める．また,このような状況における作業者の最終的なスケジュールの決定の流れを図 1,図 2 に示す．

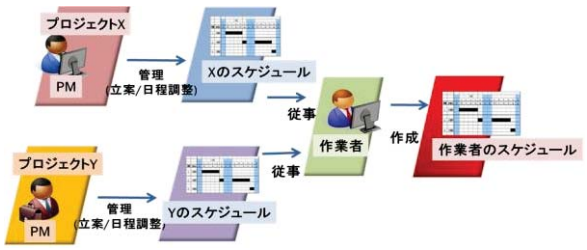

図 1: 複数のマネージャに管理される作業者のスケジュール決定の流れ 1

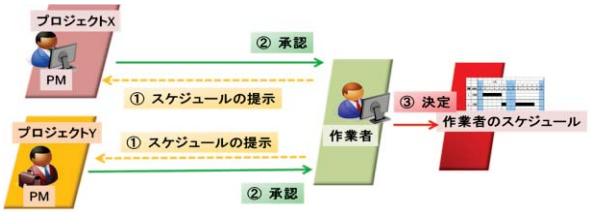

図 2: 複数のマネージャに管理される作業者のスケジュール決定の流れ 2

2. ソフトウェア開発計画問題が持つ制約

本研究では，ソフトウェア開発計画(＝ スケジュールと要員割当に関する計画)案が満たさなければならない条件を制約として捉える[2,3]．以下にソフトウェア開発計画立案問題が持つ制約の内容を具体的に記す．

2.1 作業順序に関する制約

ソフトウェア開発の各工程は，中間成果物を媒介として，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 に着手する前に生成され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．

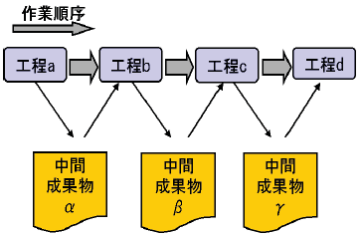

図 3：中間成果物に基づく作業の順序に関する制約

2.2 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となるスキル，資格，機能などを持つ人的リソース（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．これを，リソースの割り当て条件に関する制約と呼ぶ．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の作業を遂行する能力を持つ者でなければ担当できない．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．

2.3 リソースの割り当て期間に関する制約

ソフトウェア開発の各作業には，割り当て条件を満足するリソースでも，そのリソースにとって割り当て可能な期間（＝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がある．このような制約をリソースの割り当て可能期間に関する制約と呼ぶ．

2.4 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容量（”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働率（単位は%）の上限値で表現する．すなわち，そのリソースに割り当てられた一日の作業時間の合計（並列に進められる複数の作業に同一のリソースが割り当てられた場合にはそれらの合計)を，そのリソースが一日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，予め設定された各リソースの稼働率の上限をもって，そのリソースの能力的限界に関する制約と呼ぶ．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 つが割り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，P1 の稼働率の上限が 80%以上に設定されていれば，これらの作業は P1 に割り当て可能であるが，80%未満に設定されていれば，これらの作業は P1 には割り当て不可能となる．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り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．また，稼動率が 100%つまり 1 日に割り当てられる上限は 8 時間をデフォルトと現在している．

3. (顕在的)ボトルネック工程と潜在的ボトルネック工程の定義とその事例

(1) ボトルネック工程の定義とその事例

複数のプロジェクトで共通の人的リソース(開発要員)または非人的リソース(マシン環境など)を使用している工程があり，この工程は日程を調整することも，この工程で使用するリソースを取り替えることもできない状況にあったとする．この工程のこのような状況をボトルネック状態と呼び，ボトルネック状態にある工程をボトルネック工程と呼ぶ．図 4 にその具体例を示す．

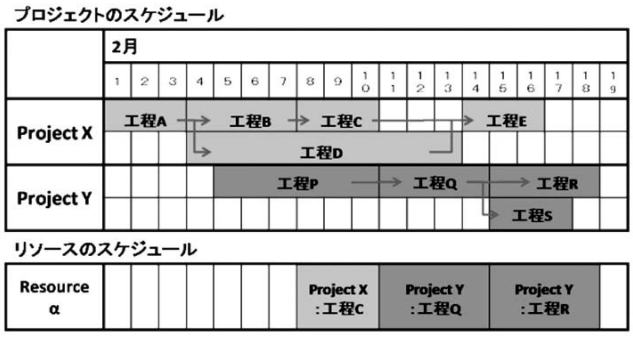

図 4：(顕在的)ボトルネック工程の例

図 4 の上部は，プロジェクトのスケジュールを表し，矢印は「作業順序に関する制約」を示している．具体的には，

図 4 の上部は「工程 A の作業が完了してから工程 B と D の作業が開始される」という作業順序に関する制約を示している．図 4 の下部はリソースの使用スケジュールを表している． 理解しやすくするために，ここでは工程 C に割り当て可能なリソースはαのみであると仮定する．

このような状況において，プロジェクト X の工程 B に遅延が発生した場合，「作業順序に関する制約」により，遅れた日数分だけ工程 C の開始日と終了日が予定より遅くなる．プロジェクト X のスケジュールだけを見ると，工程 C には３日間の余裕日があるため， ３日間の遅れまでなら，遅れた日数分だけ工程 C の開始日と終了日を遅らせることで調整できると読み取れてしまう．しかし，そのようにすると「リソースの割り当て可能期間に関する制約」により，工程 C，Q，R に割当てられているリソースαは，遅れた日数分だけ工程 Q の作業に着手するのが遅くなり，工程 Q のために３日間の作業日数を確保できなくなるので，工程 Q にはリソースαを割り当てることができない．そのため，リソースαを工程 C に割り当てるには，リソースαの空きスケジュールが３日以上連続する１９日まで待たなければならない．このため，要員の追加，プロジェクト Y 工程 Q のリソース変更などを行わない限り，プロジェクトの終了日が大幅に延期され，プロジェクトは失敗してしまう．このため，工程 C はその作業日程を１日も動かすことができない．また，工程 C に割り当て可能なリソースはαだけなので，工程 C では使用するリソースを取り替えることもできない．従って，図 4 の例では，工程 C がボトルネック工程(厳密には顕在的ボトルネック工程)である．

(2)　潜在的ボトルネック工程の定義とその事例

元来はボトルネック状態になかった工程が，先行工程の遅延の影響を受け，その実施日が遅くなることによって，ボトルネック工程に変質する場合がある．そのような状況にある工程を潜在的ボトルネック工程と呼ぶ．

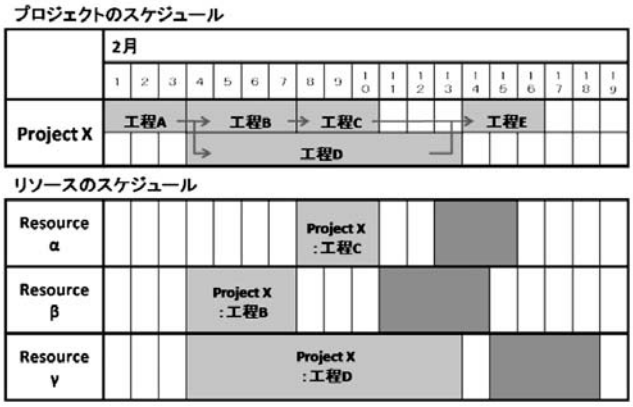

図 5：潜在的ボトルネック工程の例

図 5 を例に採って説明する．工程 B，C，D にリソースβ，α，γがそれぞれ割り当てられていてβ，α，γ以外に割当て可能なリソースは存在しないとする．また，工程 A，E に関しては，割り当てるリソースに何の制約もないので何日でも遅らせることが可能であるとする．図 26 において，工程 C に割り当て可能なリソースはαしか存在しない．しかも，リソースαの制約により，工程 C は最大で２日間までしか遅らせることができない．このため，２日間の工程遅延が工程 C に及ぶと，工程 C はその作業日程を動かすことも，使用するリソースを取り替えることもできなくなる．つまり，２日間の工程遅延で工程 C はボトルネック工程に変質する．従って，工程 C が潜在的ボトルネック工程の例である．

なお，図 5 における工程 C のように，先行工程の遅延の影響を受けなくても既にボトルネック状態にある工程を，顕在的ボトルネック工程(略称：OBN 工程)と呼び，潜在的ボトルネック工程(略称：PBN 工程)と区別する．そして，OBN 工程と PBN 工程を総称してボトルネック工程(略称：BN 工程)と呼ぶ．また，ボトルネック工程ではない工程を非ボトルネック工程(略称：非 BN 工程)と呼ぶ．

4. 研究の契機となった事例

本研究の契機となった事例を図 6 に示す．2 つのプロジェクト,プロジェクト X とプロジェクト Y の工程にリソースα,β,γ,δが割り当てられていて,α,β,γ以外のリソースは割り当て可能なリソースが存在しないとする．また,プロジェクト X の方がプロジェクト Y に対して組織的優先順序が高く,各リソースはプロジェクト X に遅れが生じないようにプロジェクト X の工程の作業を優先するものとする．加えて,プロジェクト Y の工程 o，工程 p は遂行するために特殊なスキルが必要で，特定の作業者しかできない工程であるとする．つまり,工程 o を遂行可能なのはリソースδのみ,工程 p を遂行可能なのはリソースαのめである．以上の点を踏まえて,工程 o の着手日が３日遅延した場合を考える．図 6 において工程 o は遅延してもプロジェクト完了日には影響を与えないクリティカルパス外の作業である．通常このような工程はプロジェクトを管理するプロジェクトマネージャの目が届きづらく，一見して問題が無いように見える．しかし,図 7 のリソースαのスケジュールを見ても分かるように,

次に，並行するプロジェクトの作業と競合が発生してしまうことを考える．工程 o とプロジェクト X における工程 b との間に競合が発生してしまう．発生した競合を解消するために原則(組織的優先順位の高い作業を優先する)を守り，プロジェクト X に影響が及ばないようにプロジェクト X の工程を優先するようにスケジュールを変更すると，Y の完了日が図 8 に示すように大幅に遅れてしまうことがわかる．

今回の事例から以下のような問題点が考えられる

まず，プロジェクトマネージャと作業者の２つの視点で問題点を挙げる，まず，プロジェクトマネージャの視点ではプロジェクトマネージャが作業者の状況を的確に把握できないこと他のプロジェクトからの影響を予見できないこと

が挙げられる．また，作業者の視点では作業者が複数のプロジェクトに従事することから，それを管理するプロジェクトマネージャが複数存在することになり，作業者がどのプロジェクトマネージャの指示に従ったらよいのかわからないことが挙げられる．

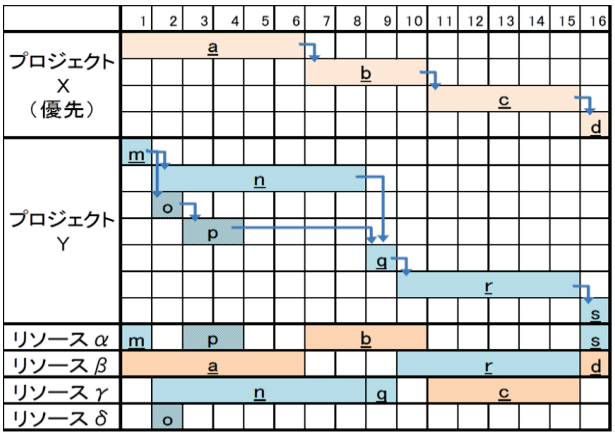

図 6：研究の契機となった事例

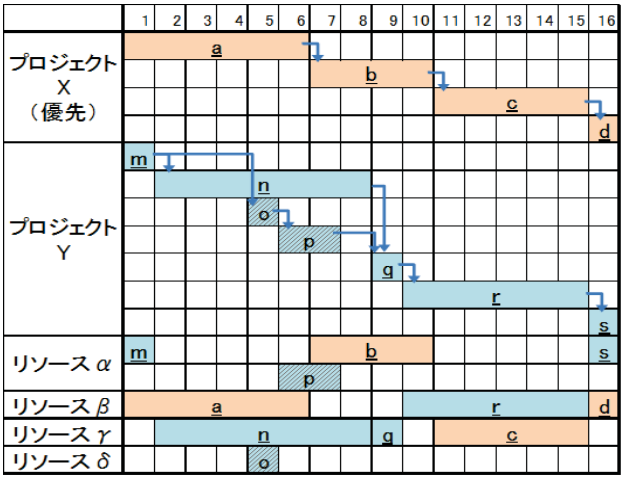

図 7：事例における競合の発生

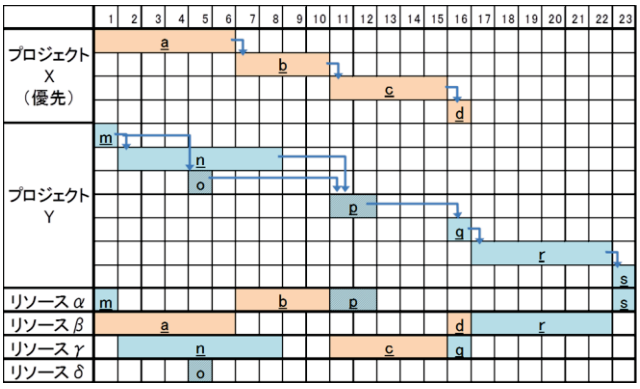

図 8：競合発生による後続工程への影響

5. 作業の競合状態

主に，複数の作業を並行して遂行する人員のスケジュール上において，もともとはスケジュールが重なってはいなかったが，スケジュールの変更を受けて担当している作業のスケジュールが重なってしまうことが頻出する．本研究においてこのような作業の競合の発生および，その影響を問題点の一つとして挙げ，競合の発生の防止および，競合が発生した場合に正しい優先順序の設定を導くことでプロジェクトの影響度を最小限にする方法を提案するために，この章では本研究で取り扱う作業の競合について明確な定義を行う．

5.1 作業の競合状態

作業の競合とは，ある作業者の着手日を迎えている工程の稼働率の合計が，リソースの容量を上回る状態にあることをいう．

リソースの能力的限界に関する制約で述べた通り，作業者は各々の容量の範囲内で，複数の工程を並行して行うことができるが，この状態は作業の競合には含まれない．また，ソフトウェア開発計画の初期計画の立案時には，前述の全ての制約を満たしている計画案を採用するため，プロジェクト開始時に作業の競合は発生しない．これはリソースが，単一のプロジェクトに割り当てられている場合でも，複数のプロジェクトを掛け持ちしている場合でも同様である．それにも関わらず，作業の競合が発生する原因としては，前工程の遅延，現在仕掛かり中の工程の遅延の２つが考えられる．プロジェクトの進捗状況は日々変化するため，具体的な作業の競合を事前に予測することは困難である．

5.2 作業の競合

作業の競合とは，ある作業者の着手日を迎えている工程の稼働率の合計が，リソースの容量を上回る状態にあることをいう．

リソースの能力的限界に関する制約で述べた通り，作業者は各々の容量の範囲内で，複数の工程を並行して行うことができるが，この状態は作業の競合には含まれない．また，ソフトウェア開発計画の初期計画の立案時には，前述の全ての制約を満たしている計画案を採用するため，プロジェクト開始時に作業の競合は発生しない．これはリソースが，単一のプロジェクトに割り当てられている場合でも，複数のプロジェクトを掛け持ちしている場合でも同様である．それにも関わらず，作業の競合が発生する原因としては，前工程の遅延，現在仕掛かり中の工程の遅延の２つが考えられる．プロジェクトの進捗状況は日々変化するため，具体的な作業の競合を事前に予測することは困難である．

研究報告用原稿の作成から投稿までの流れは，次の通りである．

5.3 作業の競合発生時の対応

作業の競合ではαは 2/8 から 2/10 までプロジェクト A の工程 C(稼働率 80%)を実施し，その後 2/11 から 2/14 はプロジェクト B の工程 Q（稼働率 80%），2/15 から 2/17 は同工程 R(稼働率 80%)を遂行することになっている（図 9）．このようなスケジュールの基で作業を進め，現在日が 2/11 に至った際に，工程 C に２日間の工程遅延が発覚した場合，αが着手日を迎えている工程は工程 C(稼働率 80%)と工程 P(稼働率 80%)の２つとなる．このときこれらの稼働率の

合計は 160%となり，αの容量 80%を超過しているため，作業の競合状態が生じる（図 10）．
が発生した場合，リソースは着手中の全ての工程を予定の期日までに完了できないことを意味する．よって，このような事態が発生した場合には速やかにマネージャにその旨を報告し，指示を仰がなければならない．また，マネージャは作業者からのこのような申告に基づいて，遅延に対する対策案の立案を行うことになる．
ここで，前述のような競合がプロジェクトをまたいで発生した場合を考えてみる．このとき作業者αは工程 C と工程 Q のどちらを優先すべきかを自分で判断できないため，作業者αはプロジェクト A のマネージャには工程 C が遅延している旨を，プロジェクト B のマネージャにはプロジェクト A の工程 C の遅延により，工程 Q への着手が遅れそうである旨をそれぞれ報告し，マネージャ同士の話し合いを行ってもらうことにより，作業の競合状態を解決する．
マネージャ間での調整では，本来は競合の対象となっている工程が，それぞれのプロジェクト全体に対してどのような影響を与えるかについて吟味し，お互いの妥協点を探すべきである．しかしながら，マネージャ同士はお互いのプロジェクトの状況をよく知らない場合が多いので，客観的な判断を下すことは難しく，プロジェクトの規模，クライアントの重要度，マネージャの声の大きさなど，プロジェクトの表層的な属性に意思決定が左右されやすいという問題がある．
本研究では組織内のプロジェクトが全体最適となるように，それぞれのプロジェクトの後続工程への影響がなるべく少なくなるような意思決定を支援する方法を提案する．

▼現在日

		2月											
		7	8	9	10	11	12	13	14	15	16	17	18
作業者α（容量=80%）	プロジェクトA		工程C(80%)										
	プロジェクトB					工程Q(80%)				工程R(80%)			
αの稼働率			80	80	80	80	80	80	80	80	80	80	

図 9：作業の競合の例題(競合発生前)

▼現在日

		2月											
		7	8	9	10	11	12	13	14	15	16	17	18
作業者α（容量=80%）	プロジェクトA		工程C(80%)										
	プロジェクトB					工程Q(80%)				工程R(80%)			
αの稼働率			80	80	80	160	160	80	80	80	80	80	

図 10：作業の競合の例題(競合発生後)

6. 後続工程への影響を考慮した作業優先順序の設定方法

6.1 後続工程への影響からみた工程の分類と略称

工程がプロジェクトの後続工程にどのような影響を及ぼすかについては，属性による工程の分類を行うことで明らかにできる．本章ではクリティカルパスとボトルネックという属性を導入して，各属性を持つかどうかによって工程を分類する．

(1) クリティカルパス
先行工程と後続工程との間に時間的にまったく余裕のない工程同士を，プロジェクトの開始から完了まで繋いで出来る，単位作業の連なる道筋をクリティカルパス（CP）と呼ぶ．クリティカルパス上にある工程を **CP 工程**，クリティカルパス上にない工程を**非クリティカルパス(非 CP)工程**と呼ぶ．クリティカルパスの性質上，CP 工程の遅延は，開発計画全体の遅延に直結する．

(2) ボトルネック
複数のプロジェクトで共通の人的リソース（開発要員）または非人的リソース（マシン環境など）を使用している工程があり，この工程は日程を調整することも，使用するリソースを取り替えることもできない状況にあったとする．このような状況をボトルネック状態と呼び，ボトルネック状態にある工程をボトルネック工程と呼ぶ．

(3) 潜在的ボトルネック
ボトルネック状態になかった工程が，先行工程の遅延の影響を受け，その実施日が遅くなることによって，ボトルネック工程に変質する場合がある．そのような工程を**潜在的ボトルネック（PBN）工程**と呼ぶ．なお，先行工程の遅延の影響を受けないでも既にボトルネック状態だと判る工程を，**顕在的ボトルネック（OBN）工程**と呼ぶ．そして，OBN 工程と PBN 工程を総称してボトルネック（BN）工程と呼ぶ．また，ボトルネック工程ではない工程を**非ボトルネック（非 BN）工程**と呼ぶ．

6.2 競合が発生した場合の優先順序決定方法

ここでは，上記に分類した工程同士が競合した場合，どのような優先順序を設定しなければならないのか分類した．工程はボトルネックかどうか（OBN 工程，PBN 工程，非 BN 工程）と，クリティカルパスかどうか（CP 工程，非 CP 工程）の 2 属性の組み合わせを持つ．工程 2 つを一対比較し，作成した 16 のルールのうち一部の導出過程を説明する．

6.2.1 CP 工程と BN 工程が競合する場合の考え方

工程と BN 工程は，その工程に遅延が生じると後続工程への影響が大きいという点ではどちらも変わらないが，これらの工程が競合し，一人の作業者がこれらの工程のどちらか一方を選んで作業をしなければならなくなったとき，どちらを優先して選ぶべきか，という問題を考えてみたい．ここで，一人の作業者が担当する２つの工程が競合するとは，一方の工程の開始予定日から終了予定日までの間に，もう一方の工程の開始予定日または終了予定日が入るようなスケジュールになっていることを意味する．

(1) **CP 工程と OBN 工程とが競合する場合の考え方**

この場合，OBN 工程は既にボトルネック状態にある．そのため，既に１日の余裕もなく直ぐに着手しなければならない状況にある．この点では CP 工程も同じである．こ

の場合には次のように考える．OBN 工程は，その定義からも分かるように，既に日程を調整することも，この工程を遂行するための要員を交代させることもできない状況にある．一方，CP 工程は１日の遅延も許されないが，この工程を遂行するための要員を交代させることは可能である．このため，競合するこれら２つの工程を担当する作業者としては，CP 工程よりも，OBN 工程を優先して選んで作業をすべきである．本稿では，このことを次のように表現する．

{OBN, CP} ⇒ OBN ＜ CP ・・・・・・ (式 1)

このとき，この作業者が OBN 工程を優先させることにより，自分が従事できなくなった CP 工程の作業を誰かに代わって貰えるように，可能な限り早く申告する必要がある．

(2) CP 工程と PBN 工程とが競合する場合の考え方

この場合，PBN 工程はボトルネック状態になるまでに未だ何日かの余裕がある．しかし，CP 工程は既に１日の余裕もなく直ぐに着手しなければならない状態にある．このとき，CP 工程の終了日の翌日まで PBN 工程への着手を送らせても，PBN 工程がボトルネック状態にならない場合には，CP 工程を優先させ，この工程の作業が終了してから PBN 工程の作業に着手すればよい．この場合の作業の優先順序は次のように表現できる．

{PBN, CP | $\mathbf{TD_{CP}} < \mathbf{TB_{PBN}}$} ⇒ CP ＜ PBN (式2)

しかし，CP 工程の終了日の翌日まで PBN 工程への着手を送らせたら，PBN 工程がボトルネック状態になるような場合には，この作業者は PBN 工程の作業を優先的に選ぶ必要がある．従って，この場合の作業の優先順序は次のように表現できる．

{PBN, CP | $\mathbf{TB_{PBN}} < \mathbf{TD_{CP}}$ } ⇒ PBN ＜ CP (式 3)

このとき，この作業者が PBN 工程を優先させることにより，自分が従事できなくなった CP 工程の作業を誰か別の要員に代わって貰えるように，可能な限り早く申告する必要がある．

CP工程とBN工程が競合する場合の考え方を表にまとめたものが下図である．ただし*は競合状態にある工程の終了を待っても OBN に変質しない事を表す．

6.2.2 工程同士が競合した場合の考え方まとめ

6.2.1 と同様にして 6.1 で分類した全ての工程の組み合わせに対して優先順序設定のためのルールの導出を行い．表にまとめたものを表 1 に示す．この表は作業の競合が発生した際の優先順序を設定するために使用する．

表 1：工程同士が競合した場合の考え方まとめ

			プロジェクトY					
			OBN		PBN		非BN	
			CP	非CP	CP	非CP	CP	非CP
プロジェクトX（優先）	OBN	CP	X	X	X	X	X	X
		非CP	X	X	X	X	X	X
	PBN	CP	Y*	Y*	X*	X*	X	X
		非CP	Y*	Y*	Y*	X*	X	X
	非BN	CP	Y	Y	Y	Y	X	X
		非CP	Y	Y	Y	Y	Y	-

6.3 事例に対する本手法の適用

4 章で述べた研究の契機となった事例に対して本手法を適用する．図 7 において発生した工程 b と工程 p に注目する．まず工程 b と工程 p を 6.1 で述べた方法で分類を行う．工程 b はプロジェクト X における CP 工程であり非 BN 工程である．また工程 p はプロジェクト Y における非 CP 工程かつ PBN 工程である．よって 6.2 より工程 b の工程の完了日(TD)が 10 日目であることと，工程 p の PBN が顕在化する日(TB)が 5 日目であることに注目して表 1 の該当箇所を参照することで式 3 より TB_PBN ＜TD_CP の条件下において PBN 工程を優先し，CP 工程を他人に交代してもらう方法をとる．よって決定した対処方法に基づいて対策案の立案を行う．工程 b の担当者をリソース α からリソース β に変更し，リソース α は予定通り工程 p の作業を遂行する．提案手法を事例に適用した結果，作業の競合が発生した両プロジェクトの完了日への影響を防ぐことができた．もしも，リソースの変更ができない状況であっても，プロジェクト X の遅延は 1 日だけで済み，組織的優先順序で決定したスケジュールに対して，十分検討すべきスケジュールの立案に成功した．

7. システムの利用イメージ

表 4 で示した判断基準はスケジュールが複雑になるにつれて，全ての作業を誤りなく行うことが難しい．特に工程遅延発生時の判断の誤りはプロジェクトの最終結果に重大な影響を及ぼす可能性がある．そこでこのような作業を作業優先順序決定支援システムとして構築し,作業者を支援する．以下にシステムの概要を述べる．

作業者はシステムから出力される優先順序の設定されたスケジュールを確認し自分の行うべきスケジュールを決定する．そして，決定したスケジュールに従って作業を遂行すことで全体最適なスケジュールの実施が保証される．しかし，もしも出力されたスケジュールにプロジェクトマネージャの承認が必要な優先順序が設定されていた場合，

作業者が最終的なスケジュールを確定するためにはプロジェクトマネージャの承認が必要となる．承認を必要とする理由は，作業間に競合が発生した場合，どちらかの作業を優先すると，優先されなかった作業にはその期間着手できず作業の属性（CP,BN など）によっては遅延が確定してしまうため,作業者個人の判断で決定してよい問題ではない．そのため作業者が取るべき最善の行動は，後続工程への影響が最小となるような対策案を，スケジュールの変更がされなかった場合より大きな被害を受けるプロジェクトのプロジェクトマネージャに提案し承認を受けることで，最終的なスケジュールが確定する．承認を行ったマネージャは競合していた別のプロジェクトのプロジェクトマネージャにスケジュールの変更に伴う遅延対策を要求する．

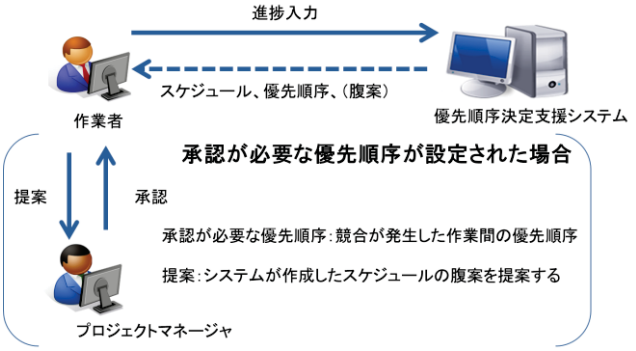

図 11：システムの利用イメージ

8. まとめと今後の課題

本稿では同時に複数のプロジェクトを兼務する作業者に対し，どの作業を優先して遂行するべきかの判断を支援するために，後続工程に及ぼす影響の大きさを考慮したタスクの優先順位付けのための考え方を述べた．今後はシステムを実装し,それを用いることで実際のプロジェクトと同程度の複雑さの例題を使ったシミュレーションを行い，支援方法の有効性について確認すると共に，実際の開発現場への適用を目指す．

参考文献

1) Project Management Institute Inc.,“プロジェクトマネジメント知識体系ガイド第 4 版（PMBOK ガイド),” 2008.
2) 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-1, Vol.J79-D-1, No.9，pp.544-557，Sep. 1996.
3) S. Komiya, and A. Hazeyama, ”A Meta-Model Framework for Software Project Management System,” IEICE Trans. Inf. Syst., Vol.E81-D, No.12, pp.1415-1428，Aug. 1998.
4) A. Hazeyama, S. Komiya, ” Workload Management Facilities for Software Project Management.” IEICE Transaction Information Systems, Vol.E81-D, No12, pp.1404-1414， Dec 1998.
5) D. Kinoshita, R. Yaegashi, H. Hashiura, K. Uenosono, and S. Komiya, ”An Automatic Schedule Planning System,” Strategies and Evaluation for Implementing the System, Joint Conference on Knowledge-Based Software Engineering 2004(JCKBSE’04), Protvino, Russia, pp.37-48，Aug. 2004.
6) 八重樫理人，木下大輔，橋浦弘明，上之薗和宏，林雄一郎，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,” 電子情報通信学会論文誌 D-1, Vol.J88-D-1，No.2，pp.215-227，Feb. 2005.
7) 高須賀公紀，嶋村彰吾，内川裕貴，木下大輔，林雄一郎，古宮誠一，“余裕日をも考慮に入れたソフトウェア開発計画の自動立案,” 情報処理学会論文誌，Vol.48，No.8，pp.2713-2724，Aug. 2007.
8) D. Kinoshita, R. Yaegashi, K. Uenosono, H. Hashiura, H. Uchikawa, and S. Komiya: "Automatic Creation of a Crashing-Based Schedule Plan as Countermeasures against Process Delay," International Jouranal of Systems Applications, Engineering and Development, Issue 4, Vo. 2, pp. 170-177, 2008.
9) 木下大輔，内川裕貴，小坂祐也，古宮誠一: “摂動を用いた影響波及解析によるボトルネック工程の自動検出,” 情報処理学会論文誌, Vol.50, No.12, pp. 3084-3094, Dec. 2009.