Vol.2011-SE-171 No.18
2011/3/15

複数のプロジェクトに属する作業者の作業優先順序の自動設定

比護　徹治[†]　木下　大輔[†††]　小坂　祐也[††]　古宮　誠一[††]

企業ではソフトウェア開発を行うに当たり短納期で開発を行うために、複数のプロジェクトを同時期に行うことが一般的になってきた。そのため、各作業で必要となる作業要員が複数のプロジェクトに共有される。これによって作業者は自分の所属するプロジェクトの各作業に優先順序を付けて、時間配分をして作業を進める必要がある。しかし、この場合、作業に優先順序を決定することが容易でないことが多い。なぜならば、作業者が優先順序を決定するためには作業者が所属する各プロジェクト、各部署における作業の情報とその関連をすべてチェックしたうえで行わなければならない。そのチェックしなければならない情報が非常に膨大で、考慮不足などにより誤った優先順序の設定を行ってしまうと、重要な作業に遅延が生じればプロジェクト全体にまで影響が波及してしまう。そういったリスクに対処するためんに優先順序の設定を人の頭の中だけで行うのではなくシステムによって自動的に設定を行い作業者の優先順序の設定を支援することが本研究の主題である。
また、古宮、小坂らによって作業優先順序自動設定システムにおける手法が提案されているが、本研究ではそれに不足した手法であるボトルネック工程を考慮した作業優先順序の設定手法を提案する。また、先行研究で提案された手法に従って作業の優先順序の設定を行うと、作業者個人では優先順序の決定ができない場合が生じてくる、そのような特殊なケースにおける優先順序の設定手法を提案する。

Automatically Prioritizing Each Task of the Worker Who Belongs to Two or More Projects during the Same Period

TETSUHARU HIGO[†]　DAISUKE KINOSITA [†††]
YUYA KOSAKA[††]　SEIICHI KOMIYA[††]

Doing two or more projects in the enterprise at a simultaneous period when the software development is done to development by the quick turn has become general. Therefore, the work force needed because of each work is shared in two or more projects. As a result, the worker should distribute putting the priority order on each work of the project that I belong at time and proceed work. However, it is not easy to decide the priority order to work in this case. Because after all information and the relation of work in each project and each post to which the worker belongs are checked, it is necessary to do so that the worker may decide the priority order. The influence spreads even to the whole project if the delay is caused in the heavy-duty work if the priority order very huge information should, and mistaking the check due to consideration shortage etc. is set. It is a subject of the present study in not straightening for the doing system to set automatically and to support the setting of the order of giving priority to the worker only by person's head that deals with such a risk as for the setting of the priority order.
Moreover, it proposes a set technique of the order of giving priority to work to consider the bottleneck process that is a technique in the present study insufficient in it though the technique in the automatic work priority order setting system is proposed by komiya and ksaka. Moreover, if the order of giving priority to work is set according to the technique proposed by the previous work, it proposes a set technique of the priority order in such a special case where the case where the worker individual cannot decide the priority order is caused.

Vol.2011-SE-171 No.18
2011/3/15

1. はじめに *

近年ソフトウェア開発はプロジェクトを組んで行われるのが一般的である。プロジェクトとは必要な人材、資材、費用などを確保するために、ある限られた期間だけ非定常的な組織を作って非定常な業務を行うものである。ソフトウェアプロジェクトでは開発のための作業スケジュールや各作業への作業要員の割り当てなどの計画を記した開発計画というものが存在する。この開発計画に従って作業者が割り当てられた作業を行うことで、プロジェクトは進む。また、
企業ではソフトウェア開発を行うに当たり短納期で開発を行うために、複数のプロジェクトを同時期に行うことが一般的になってきた。そのため、各作業で必要となる作業要員が複数のプロジェクトに共有される。これによって作業者は自分の所属するプロジェクトの各作業に優先順序を付けて、時間配分をして作業を進める必要がある。しかし、この場合、作業に優先順序を決定することが容易でないことが多い。なぜならば、作業者が優先順序を決定するためには作業者が所属する各プロジェクト、各部署における作業の情報とその関連をすべてチェックしたうえで行わなければならない。そのチェックしなければならない情報が非常に膨大で、考慮不足などにより誤った優先順序の設定を行ってしまうと、重要な作業に遅延が生じればプロジェクト全体にまで影響が波及してしまう。そういったリスクに対処するためんに優先順序の設定を人の頭の中だけで行うのではなくシステムによって自動的に設定を行い作業者の優先順序の設定を支援することが本研究の主題である。
また、古宮、小坂らによって作業優先順序自動設定システムにおける手法が提案されているが、本研究ではそれに不足した手法である「ボトルネック工程を考慮した作業優先順序の自動設定」提案する。また、先行研究で提案された手法に従って作業の優先順序の設定を行うと､作業者個人では優先順序の決定ができない場合が生じてくる、そのような特殊なケースにおける優先順序の設定手法を提案する。
本校の構成を以下に示す。まず 2 章でソフトウェア開発計画立案問題が持つ制約とその内容について述べる。3 章ではボトルネック工程と潜在的ボトルネック工程の定義を述べるとともにその事例を示す。

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案が満たさなければならない条件を制約として捉える[1,2]．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．

（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．

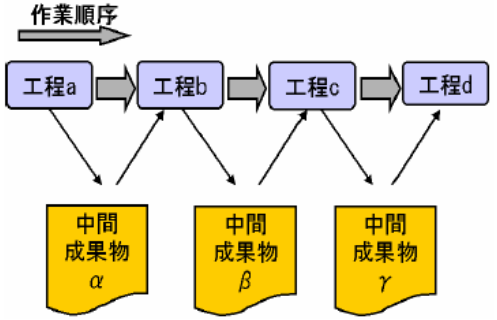

図 1 中間成果物に基づく作業の順序に関する制約

（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．

（３） リソースの割り当て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．

（４） リソースの能力的限界に関する制約

*† 芝浦工業大学
†† 芝浦工業大学大学院
†††(株)日立製作所
Hitachi Ltd.

Vol.2011-SE-171 No.18
2011/3/15

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．

プロジェクトの各工程に割り当てられたリソースの組み合わせが，制約を全て満たしていれば，その組み合わせはソフトウェア開発計画の候補と考えることができる．つまり，ソフトウェア開発計画を立案することは，制約の多い組み合わせ最適化問題を解くことであると言える．

3. 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使用しているリソースがあったとする。このリソースを使用している工程に遅延が波及したときに，その工程の日程を調整することも，そ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場合がある．このような状況にある工程をボトルネック工程と呼ぶ。図２にその具体例を示す.

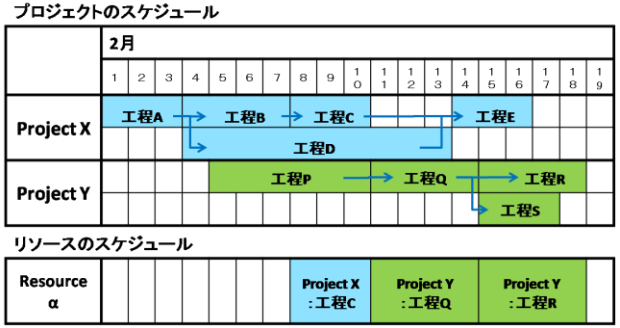

図 2　ボトルネック工程の例

図２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２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２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。図２はリソースαがプロジェクト X の工程 C と，プロジェクト Y の工程 Q と工程 R に割当てられている状況を示している．この事例では，リソースαが割り当てられている，連続する３つの工程 C，Q，R の間には余裕日がまったくないことに読者は注意して欲しい．

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２の例では，工程 C がボトルネック工程である。

プロジェクトを成功へ導くためには，たとい１日の工程遅延でもその影響が，ボトルネック工程に波及しないようにプロジェクトを進めて行かねばならない．そのためには，ボトルネック工程（図２の例では工程 C）の直前の工程（図２の例では工程 B）の終了時までに，１日の遅延も生じていない状態にしなければならない．

(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工程ではなかった工程が，工程遅延の影響を受け，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。

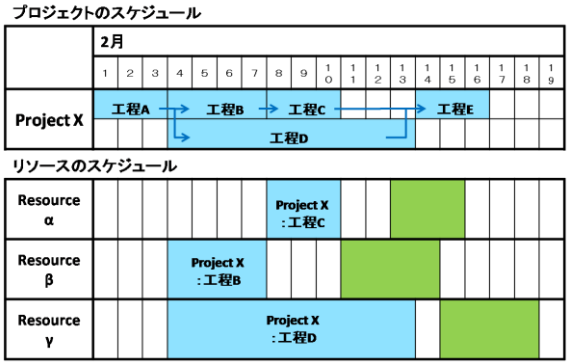

図 3　ボトルネック工程の例 2

図３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３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。

工程 B に割り当て可能なリソースはβしか存在しない。しかも，リソースβの制約により，工程 B は最大 3 日間まで遅らせることが可能であるように見える。しかし，工程 B の後続工程 C が 2 日間の工程遅延でボトルネック工程に変質するので，２日間の工程遅延で工程 B はその作業日程を変更することも，使用するリソースを取り替えることもできなくなる。つまり，２日間の工程遅延で工程 B はボトルネック工程に変質する。

工程 A の後続工程は工程 B と工程 D の 2 つである．工程 A の後続工程 B が 2 日間の工程遅延でボトルネック工程に変質するので，工程 A－工程 B－工程 C-工程 E のパスでは，工程 A は最大で 2 日間の工程遅延しか許されないことが分かる。一方，工程 A－工程Ｄを調べてみると，工程 A は割り当てるリソースに何の制約もないのでボトルネック工程にはなり得ない。これに対して，工程Ｄに割り当て可能なリソースはγしか存在しない。しかも，リソースγの制約により，工程Ｄは１日しか遅らせることはできない。このため，１日の工程遅延で工程Ｄ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，１日の工程遅延で工程Ｄはボトルネック工程に変質する。このため，工程 A－工程Ｄのパスでは，工程Ｄの先行工程Ａは１日の工程遅延しか許されないことが分かる。両者を比較することにより，工程Ａは１日の工程遅延しか許されないと結論される。

以上の考察により，図３の例では，潜在的ボトルネック工程は工程 B，工程 C，工程 D の３つであり，工程 B は２日間の工程遅延で，工程 C は２日間の工程遅延で，工程 D は１日の工程遅延でそれぞれボトルネック工程に変質する。

4. 本研究における作業の分類

組織に所属する作業者の仕事は「プロジェクト内作業」と「プロジェクト外作業」の 2 つに分類できる。プロジェクト内作業とは組織が進めているプロジェクトにおける作業のことで、対応するプロジェクトのスケジュールに記述されている作業のことである。また、プロジェクト外作業とは、プロジェクト内作業には含まれない作業のことで､通常プロジェクトのスケジュールに記載されることのない作業のことを指す。例えば、作業者の所属する課での業務、定例会議などはプロジェクト外作業である。さらに、5.1.1 節と 5.1.2 節で細かく分類し定義する。

4.1 プロジェクト内作業

各プロジェクトの線表（スケジュール）に記載されている各プロジェクトにおける作業のことで、本研究では、作業優先順序を設定するために、ボトルネック工程、クリティカルパス上作業、クリティカルパス外作業の 3 つに分類する。クリティカルパスに含まれる作業をクリティカルパス上作業と定義する。また、クリティカルパスに含まれない作業をクリティカルパス外作業と定義する。

4.2 プロジェクト外作業

本研究では、作業優先順序を設定するために、プロジェクト外作業を通常業務、定例業務、緊急業務の 3 つに分類し以下のように定義する。

・通常業務
　期限が定められており、作業者が工数を見積もって作業する業務
　(例　マネジメント作業、備品チェック等)
・定例業務
　定期的に行うことが決まっていて、具体的な実施日時があらかじめ決められている業務
　(例　課のミーティング、進捗会議など)
・緊急業務
　業務の遂行指示を受けてから即座に対処が必要な業務
　(例　上司からの検討依頼、システムトラブルなど)

また、緊急業務を緊急業務(ボトルネック扱い)、緊急業務(非ボトルネック扱い)の２つに分類する。前者は緊急性が高くかつ、他の人員を割り当てることができないような作業で、後者は緊急性は高いが、他の人員を割り当てることができる作業とする。

5. 作業優先順序の設定

本研究における作業優先順序の設定方法は大きく分けて 2 通りあり、この章では、通常時における作業優先順序の設定方法について述べ、6 章において特殊なケースにおける作業優先順序の設定手法について説明する。

　通常時は作業の余裕時間を算出し、余裕時間の少ない作業で優先順序の設定を行う「余裕時間に基づく優先順序の設定」を用いる。

4.1 余裕時間に基づく優先順序の設定

　余裕時間に基づく優先順序とは，各作業について期日までにどれだけその作業以外の作業に時間を割けるかを「余裕時間」と定義し，それによって表わされる優先順序である．

まずは，現在の日付 D_p から期日 D_l までにその作業を進めることが可能な時間（作業可能時間と定義する）を求める．各作業の作業可能時間 T_a は以下の式で求められる．ただし，1 日の作業可能時間を T_d ，期間内の最優先作業の合計作業時間を T_p とする．

$$T_a = \left(D_l - D_p\right) \times T_d - T_p \tag{1}$$

次に，求めた作業可能時間と作業時間見積もりを用いてその作業の余裕時間を求める．作業の余裕時間 T は，以下のように定義される．ただし，作業の進捗（百分率）を P ，作業時間見積もりを T_e とする．

$$T = T_a - T_e \times \left(1 - P\right) \tag{2}$$

この余裕時間が少ない作業から順に，余裕時間に基づく優先順序を高く設定する．

6. 作業優先順序の設定における特殊なケース

　この章では作業者が並行して作業を行う際に、作業の競合（作業時間の関係上遂行不可能な作業が生じてしまうこと）が生じた際の優先順序の設定方法を述べる。

　優先順序の設定の際に、作業者個人では判断が難しい、または不可能であるようなケースが生じることがある。作業が競合、つまりどちらかの作業しか遂行できないようなとき作業者は正しく判断できないので、管理者に連絡し指示を仰ぐ必要がある。このとき競合する作業の関係で取るべき対応が異なるので、図 4 のように作業間の優先順序を定め、それに従って 6.1～6.3 節でそれぞれのケースについて説明する。

Vol.2011-SE-171 No.18
2011/3/15

	緊急業務（BN扱い）	緊急業務（非BN扱い）	定例業務	通常業務	ボトルネック工程	クリティカルパス上作業	クリティカルパス外作業
緊急業務（BN扱い）		高	高	高	要相談	高	高
緊急業務（非BN扱い）	低		要相談	要相談	低（要報告）	低（要報告）	高
定例業務	低	要相談		要相談	低（要報告）	低（要報告）	高
通常業務	低	要相談	要相談		低（要報告）	低（要報告）	自分で判断
ボトルネック工程	要相談	高（要報告）	高（要報告）	高（要報告）		高	高
クリティカルパス上作業	低	高（要報告）	高（要報告）	高（要報告）	低		高
クリティカルパス外作業	低	低	低	自分で判断	低	低	

図 4　作業間の優先順序

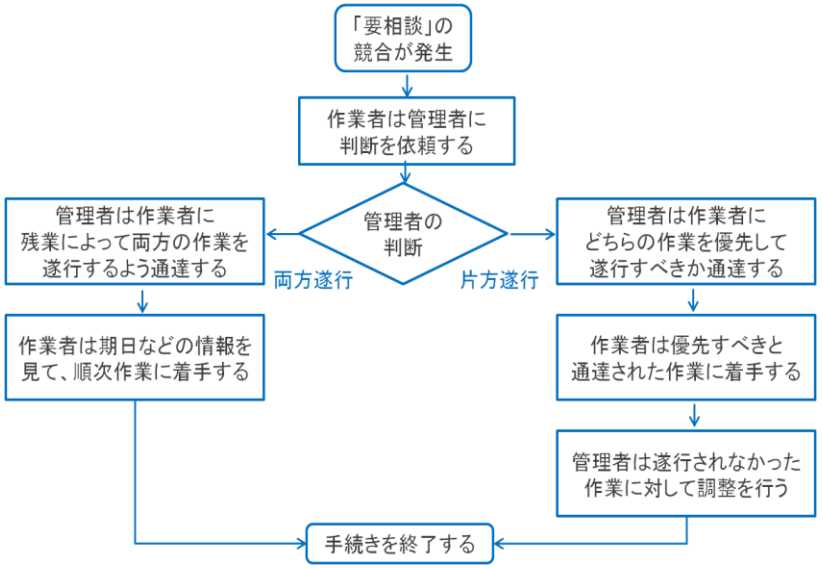

図 5　「要相談」のフローチャート

4.1　作業優先順序が「要相談」となるケース

図.5 より優先順序が「要相談」となるときは、図.5 のフローチャートに従って判断すれば、正しい優先順序の設定が可能となる。また、優先順序が「要相談」となるのは以下の競合が発生した場合である。

・ボトルネック工程と緊急業務(ボトルネック扱い)競合した場合
　どちらも変わりの人員がいないので、どちらかをあきらめなければならない。ただし、残業などで両方遂行する場合もある。
・緊急業務(ボトルネック扱い)以外のプロジェクト外作業の同士が競合した場合
　どちらも人員の変更が可能であるが、判断は管理者にしかできないため、管理者に判断を依頼する。

4.2　作業優先順序が「要相談」となるケース

図 6.より優先順序が「要報告」となるとき、図.6 のフローチャートに従って判断すれば、正しい優先順序の設定が可能となる。また、優先順序が「要報告」となるのは以下の場合である。

・「緊急業務(非ボトルネック扱い)・定例業務」と「ボトルネック工程・クリティカルパス上の作業」が競合した場合
　作業者は作業間の優先順序に従って作業を遂行する。作業を遂行していく中で優先順序の関係でプロジェクト外作業の遂行が不可能であることが分かったならば、作業者は管理者に報告する。もし、管理者がプロジェクト外作業を優先させたいと判断したならば、管理者が作業者に優先する作業の変更を通達する。

Vol.2011-SE-171 No.18
2011/3/15

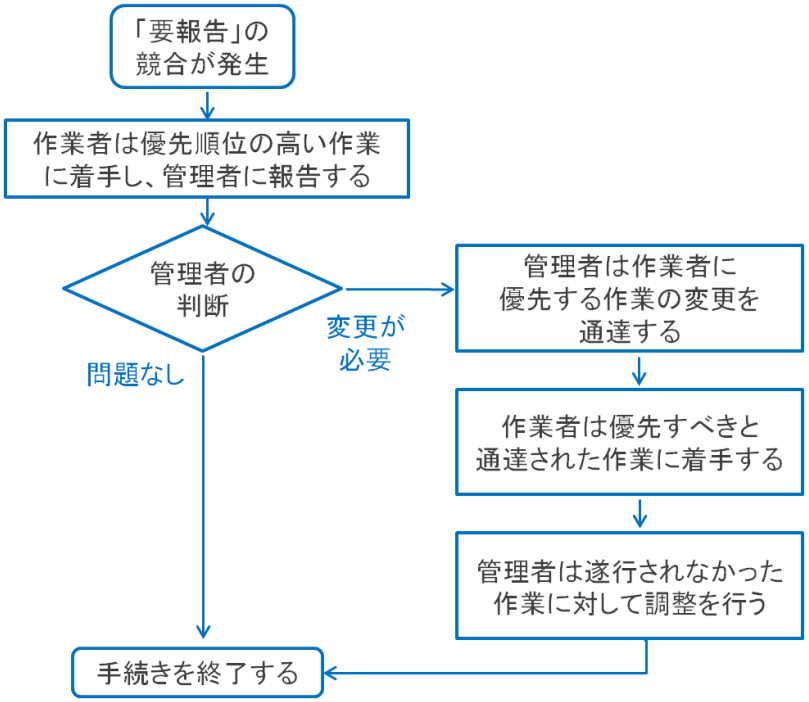

図 6　「要報告」のフローチャート

4.3 作業優先順序が「要相談」となるケース

図.7 より優先順序が「自分で判断」となるとき、図.7 のフローチャートに従って判断すれば、正しい優先順序の設定が可能となる。また、優先順序が「自分で判断」となるのは以下の場合である。

・「クリティカルパス外作業」と「通常業務」が競合している場合
優先順序は同程度なので、余裕時間をみて作業者が自分で判断する。もしも、作業期限までに終わらない作業が発生したならば、即座に管理者に報告する。

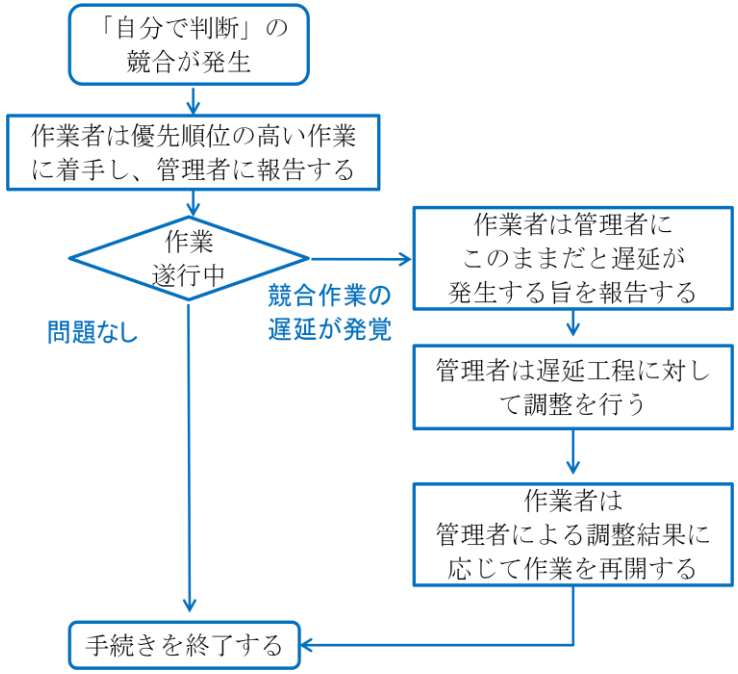

図 7　「自分で判断」フローチャート

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 TranINF SYST, vol.E81-D.No12, pp1415-1428, Dec 1998
3) A. Hazeyama, S. Komiya, “Workload Management Facilities for Software Project Management,” IEICE Tran INF SYST, vol.E81-D.No12 , pp1404-1414, Dec 1998.
4) R. Yaegashi, D. Kinoshita, H. Hashiura, K. Uenosono, S. Komiya, "Automatically Creating a Schedule Plan as Countermeasure by Means of "Crashing" against Process Delay," JCKBSE'04, pp.24-36, Protvino, Russia, Aug. 2004.
5) 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-I, Vol. J88-D-I, No.2, pp.215-227, 2005.

6) Leach, Lawrence P., “Critical Chain Project Management”, Artech House Professional Development Library, London 313, 2000
7) M. H. Penedo and E. D. Stuckle, “PMDB - A project master database for software engineering environments,” 8th International Conference on Software Engineering, pp.150-157, 1985.
8) L. Liu and E. Horowitz, “A formal model for software project management,” IEEE Trans. on Software Engineering, vol.15, no.10, pp.1280-1293, Oct. 1989.
9) L .Liu and E. Horowitz, “Object database support for a software project management environment,” ACM SIGSOFT Software Engineering Notes, vol.13, no.5, pp.85-96, Nov.1988.
10) Y. Matsumoto and T. Ajisaka, “A data model in the software project database KyotoDB,” JSSST Advances in Software Science and Technology 2, pp103-121, 1990.
11) H. Sato, ”Project management expert system,” Proc.ACM CSC’87, Feb.1987.
12) D. Kinoshita, R. Yaegashi, H. Hashiura, K. Uenosono, S. Komiya, “An Automatic Schedule Planning System: Strategies and Evaluation for Implementing the System,” Joint Conference on Knowledge-Based Software Engineering 2004 (JCKBSE’04), pp.37-48, Protvino, Russia, Aug.2004.
13) 稲垣公夫, TOC クリティカル・チェーン革命，日本能率協会マネジメントセンター, 1998.
14) エリヤフ・ゴールドラット著，ザ・ゴール，ダイヤモンド社, 2001.
15) エリヤフ・ゴールドラット著，ザ・ゴール 2，ダイヤモンド社, 2002.
16) エリヤフ・ゴールドラット著, クリティカルチェーン，ダイヤモンド社, 2003.
17) 中嶋秀隆, 津曲公二, PM プロジェクト・マネジメント クリティカルチェーン，日本能率協会マネジメントセンター, 2003.
18) 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
19) Daisuke Kinoshita, Rhito Yaegashi, Kazuhiro Uenosono, Hiroaki Hashiura, Hiroki Uchikawa, and Seiichi Komiya, "Automatic Creation of a Crashing-Based Schedule Plan as Countermeasures against Process Delay"INTERNATIONAL JOURNAL of SYSTEMS APPLICATIONS, ENGINEERING & DEVELOPMENTIssue 4, Volume 2, 2008, pp. 170-177.

付録

謝辞 MS-Word のテンプレートファイルの作成にご協力頂いた皆様に，謹んで感謝の意を表する．