Tri-it

Takenouchi Award

第３回　ＩＴ懸賞論文

～・～・～・～・～・～・～・～・～・～・～・～・～・～・～
・～・～・～・～・～・～・～・～・～・～・～・～・～・～・

主催　（株）竹野内情報工学研究所

共催　（株）テレコム和歌山

協賛　（財）和歌山会社経済研究所
（財）京都産業情報センター
ＮＴＴ西日本（株）大阪支店
ＮＴＴ西日本（株）和歌山支店
（株）ウェブソリューション
ＮＰＯふぁーむいん紀州日高

後援　関西情報技術士会
上級ＳＥ教育研究会

・～・～・～・～・～・～・～・～・～・～・～・～・～・～・
～・～・～・～・～・～・～・～・～・～・～・～・～・～・～

実例に学ぶ危機への対処法

ソフトウェア開発におけるプロジェクト・マネジメント

技術士
堀上　明
(akira.horikami@nifty.com)

目　次

Tri-it

１．　はじめに

ソフトウェア開発におけるプロジェクトで、開始から終了まで何も問題が起こらない、ということは経験上まずありえない。

プロジェクト期間の長短に関わらず何らかの問題が間違いなく発生する。

また、二つとして同じプロジェクトは存在しないものである。別々のプロジェクトで同じような問題が発生した場合、あるプロジェクトに有効であった施策が、必ずしも別のプロジェクトに有効であるとは限らない。

プロジェクトの進行過程で問題が発生した時に、その対応を誤るとシステムのカット・オーバーそのものに影響を及ぼし、文字通り“危機”となる場合がある。

プロジェクト・マネージャやプロジェクト・リーダーの立場からすると、問題は発生しないに越したことはない。しかし逆に、問題があるということはプロジェクトが正常に進行している証である、と前向きに捉え、正面から問題に取り組む姿勢が大切である。

本論文では、プロジェクトが危機に直面した時にどのように解決していくべきかを、私が実際に経験したプロジェクトの実例を通して考察する。

なお、本論文の内容は私の実際の経験に基づいたものであり、私の知る限り、他者の著作権等に抵触しないことを付け加えておく。

２．本論

私は１９８７年に大学を卒業後、独立系大手ＳＩベンダーに就職し、現在も在籍している。

最初の２年はプログラマとして、主に生産管理業務のアプリケーションプログラムを開発していたが、入社３年目のころから、プロジェクト・リーダーとして、大小様々なプロジェクトを任されるようになった。

今まで、あまり大きな失敗をした経験はないが、最初から最後まで何も問題なく終了することのできたプロジェクトもほとんどない。

程度の差はあるが、必ず何らかの問題が発生し、都度対応することを余儀なくされてきた。　中には、「もし、あの時、あの手を打っていなかったら取り返しのつかない事態になっていた」と思うプロジェクトもいくつかある。

大きな書店に行けば、プロジェクト・マネジメントに関する書籍は何冊も書棚に並んでいるが、実例をもとに書かれている書籍は以外と

少ない。
そこで、私が実際に担当したプロジェクトの中で、特に印象に残っているプロジェクトの中から３つの実例をあげ、そこから得られたプロジェクトにおける危機への対処法について述べようと思う。

２．１　トラブルプロジェクトにおける状況分析

最初の事例は、ある顧客の基幹システム構築プロジェクトである。当社の作業範囲は主として、プログラム設計から結合テストまでの下流フェーズであったため、当初は若手ＳＥがプロジェクト・リーダーを任されていた。
ところが、プロジェクトが開始されてから２ヶ月ほど経つと、進捗状況が見えなくなっていた。プロジェクト・リーダーから彼の上司に行われる毎週の進捗報告では、機能別の遅れ・進みはわかるのだが、それがプロジェクト全体の進捗に与える影響度や、どのような問題点があるのかを、外部からはチェックできなくなっていた。
ただ、そのまま放置しておくと、計画したスケジュール通りにプロジェクトを終了することができないということは明らかであった。
そこで、状況把握と事態収拾のために、私が途中から参画することとなった。
私はまず、プロジェクト全体の状況を把握するために、進捗状況の確認を行った。
ところが進捗管理台帳を見ても、機能ごとに作業の開始・終了の予定日と実績日が記入されているだけで、進捗が遅れている機能について、遅延の理由、遅延の程度、終了の見込み等がわからなかった。
さらに、進捗管理台帳への進捗状況の反映も日々行われているわけではなく、プロジェクトの最新の進捗状況は進捗管理台帳を見ただけではわからなかった。
そこで、２００近い全機能の進捗状況を調査した結果、判明した問題点は以下の通りであった。

（１）プロジェクト管理上の問題点
①若手プロジェクト・リーダーが、プロジェクト管理以外に、プログラム設計や開発環境設定等、プロジェクトを進めていく上で重要な作業を、全て中心になって行っていた。
その結果、負荷が集中してオーバーフローしており、本来彼が行うべき、プロジェクト管理作業ができなくなっていた。

（２）進捗上の問題点
①顧客からの仕様提示が全体的に予定よりも遅れ気味となっていた。

②そのため、メンバーの中には進捗が遅れているにも関わらず、手待ちの状態となっている者もおり、作業効率が悪くなっていた。
③プログラム開発の見積もり工数が２人月の機能が２機能あり、それを１ヶ月強で開発しなくてはならなくなっていた。

このような状況では、すぐに手を打つ必要があると考え、対策を検討した。
（１）のプロジェクト管理上の問題点では、若手プロジェクト・リーダーが、これまでプロジェクト管理の経験がなかったにも関わらず、まわりから十分なフォローを受けていないことが原因のひとつとなっていた。そのため、彼自身がプロジェクトの何をどのようにコントロールしてよいかわからない状態になっており、プロジェクトの管理以外に、顧客との仕様調整や開発環境設定など、重要な作業を全て一人で抱え込んでいた。
要員育成という観点から、彼には引き続きプロジェクト・リーダーとして、プロジェクト全体を指揮する立場に残ってもらい、私がフォローをしていくという方法も検討した。しかし、（２）の進捗上の問題点があり、一刻の猶予も許されない状況にあった。
そのため、替わって私がプロジェクト・リーダーを担当し、各メンバーに対するプログラム設計上のフォローを彼に任せることにした。

（２）の進捗上の問題点では、顧客に対して早急に仕様提示を急いで頂くよう依頼した。また、プログラム設計を担当しているメンバーには、後工程で行うことになっていた作業のうち、顧客からの仕様の提示がなくても着手可能な作業を抽出し、前倒しで行うよう指示し、少しでも負荷を分散できるように考慮した。
また（２）③の問題に対しては、メンバー内での開発をあきらめ、開発生産性と品質面で信頼のおける協力会社に開発を依頼し、何とか対応していただくことができた。

上記の対策をとり、一部のメンバーには、深夜まで時間外作業をお願いすることにはなってしまったが、予定納期通りに開発を終了させることができた。

２．２　顧客との交渉

ある年の７月に始まったプロジェクトで抱えていた問題は、要員に関するものであった。

当時、私はある顧客に常駐し、基幹システムの保守と開発を行っていた。

その年の3月から4月にかけてベテラン社員2名が相次いで退職し、顧客の業務・システムを知っているのは私一人、という状況になっていた。

5月には、将来のキーマンとして活躍してもらうことを前提に、2名の若手メンバーを増員した。ともに優秀であったのだが、先の2名のベテラン社員が何年もかけて身につけた、顧客の基幹業務とシステムのノウハウを、彼らがすぐに理解できるものではなかった。

また、7月からプロジェクトが始まると、さらに新しいメンバーを増員することとなり、私の管理負荷はピークに達していた。

プロジェクトは主に既存システムの改定で、既存のデータベース，帳票やオンライン画面の修正が中心であり、改定内容もあまり難しいものではなく、顧客の業務・システムを熟知している者がテストを行えば、大して工数はかからないはずであった。

しかし､顧客の業務･システムを知っている人間が私一人の状況で、一人一人のメンバーに顧客の業務・システムを説明しながらの開発は遅々として進まなかった。やっとできあがったプログラムの中には、必要なテストケースを網羅しておらず、何度もテストのやり直しを指示したものがあった。

以上のような状況を顧客に説明し、スケジュール調整や、担当機能の削減等を依頼したが、元々顧客には非がないだけに、調整は難航した。

また､顧客から開発依頼のあった機能の中には、私自身が知らない業務システムの改定も含まれており、見積もり工数が顧客との間で食い違っていた。そのため、その機能については、そのまま依頼通りに開発を受けるわけにはいかず、顧客との調整が進まず着手できない状態にあった。

さらに、顧客から開発を当初の終了予定よりも一週間前倒しで終わらせるよう、依頼があった。これは、こちらが提示していた開発スケジュールでは、万一プログラムの不具合やスケジュール遅延が発生した場合に対応できないであろう、との配慮からであった。

つまり、それだけ顧客に心配をかけてしまったことになる。

すでにメンバーは相当の時間外勤務をしており、そのままの体制では、前倒しで開発を終了させることができないことが確実となった。

対応策を検討した結果、以下の3案を顧客に提示した。

①同じ顧客の、別プロジェクトで作業している当社メンバーに応援を依頼する。

②既存メンバーには、時間外・休日出勤で対応してもらう。

③私が知らない業務システムの改定については、顧客に引き取ってい

ただく。

特に③については、顧客に無理をお願いせざるを得なかった。
こちらとしても、できる限りの対策を検討した結果であることを、顧客に認識していただき、了解を得ることができた。
その後も、顧客参加のシステムテストで、設計不備によるトラブルが発生するなど、必ずしも順調に推移した訳ではなかった。しかし、最終的には予定通り開発を終了させることができた。
なお、③で顧客に引き取っていただいた機能については、顧客側の当該システムをよく知っているメンバーに対応していただいたことを補足しておく。

２．３　増員のタイミング

最後にあげる事例は、ある年の１０月から翌年５月にかけて担当したプロジェクトである。
このプロジェクトは、新聞発表でカット・オーバーの日程が公表されていたにもかかわらず、顧客側の都合でシステム要件がなかなか決まらなかったため、要員調整に苦労したプロジェクトであった。
システム要件が決まらない以上、こちらとしても必要最小限の要員を手配することしかできず、最終的に要件が確定した１１月中ごろの段階では、１月までの単体開発のフェーズで、約１０人月程度分の要員が不足することが判明した。
残りの期間がないという理由から、過去に当該顧客のシステム開発を経験したことのあるメンバーを中心に増員の手配を行った。しかし、既に他のプロジェクトに従事しており、経験者の増員は不可能であった。
そのため、顧客のシステム開発の未経験者をアサインすることを検討したが、２ヶ月という短期間では、こちらの管理負荷が増え、却って開発生産性が落ちるのは明らかであった。
やむなく、既存のメンバーだけで乗り切ることができないかを検討した。しかし、時間外、土日出勤、年末年始の休暇を返上してメンバーに作業してもらっても、どうしても３人月足りないことがわかった。
また当時は、様々な事情から、納期延長を顧客に依頼することも難しい状況だった。
要員体制表と、カレンダーを何度もにらむうちに、他のプロジェクトに従事している当該顧客のシステム開発経験者に、１２月、１月の土日と年末年始の休暇中、応援に来てもらってはどうか、ということに気がついた。

そうすれば、年末年始の休暇と、その前後数週間の土日で約１５日分の工数を捻出でき、３～４人の応援で不足工数をまかなうことができる。

応援依頼先は他部署にわたるため、早速上司に相談し要員の手配を行った。

応援を依頼するには、時期的にこれ以上手配が遅れたら、いくら応援に来てもらっても間に合わない、というギリギリのタイミングだった。

「堀上が困っているのなら」ということで、最終的には３名の応援を得ることができ、無事に開発を終了させることができた。

３．まとめ

「２　本論」で述べた３つの事例において、危機への対処という観点から、共通する点は、２つある。

プロジェクトの状況に対する冷静な「状況分析」と、「最後まであきらめない姿勢」である。

「２．１　トラブルプロジェクトにおける状況分析」では、途中からプロジェクトに参画し、プロジェクト状況の全体像の把握から、問題点に対処するまでの過程を述べた。

いわゆる”火消し”としてプロジェクトに参画した場合、問題の直接の当事者ではないので、内部から客観的にプロジェクトの状況を分析することが可能となる。

そのため、洗い出された問題点に対してもあきらめずに自信を持って対処することができる。

「２．２　顧客との交渉」においては、ベテラン社員の退職によってチーム力が低下した中、どのように問題に対処したかを述べた。

顧客からの依頼事項に対して、できることとできないことを判断し、対応できない部分に関して、顧客にねばり強く説明し、スケジュール調整や機能の引き取りをお願いした。

結果として顧客に無理をお願いしたことになってしまったのは残念ではあったが、出来る限りの対策を検討し、誠意を持って対応した結果、顧客の了解を得ることができたのである。

「２．３　増員のタイミング」では、納期延長も増員もままならない状況で、どのように問題に対処したかを述べた。

年末年始の休暇期間に、他のプロジェクトから応援に来てもらう、というのは、決してほめられる対策ではない。しかし、当時の状況から見て、他にそれ以上の手段がなかったのも事実である。

残りの開発期間と、投入可能な工数を比較し、あきらめずに検討し

てやっとのことで、導き出すことができた対策である。

プロジェクト・マネジメントの手法に関しては、多くの書籍に述べられているが、それらを実際の生きたプロジェクトに応用し、成功させるためには、「状況分析」と「最後まであきらめない姿勢」がとても大切である。

4 参考文献

（1）「実践的ソフトウェア開発工程管理」竹山　寛
技術評論社　２０００年