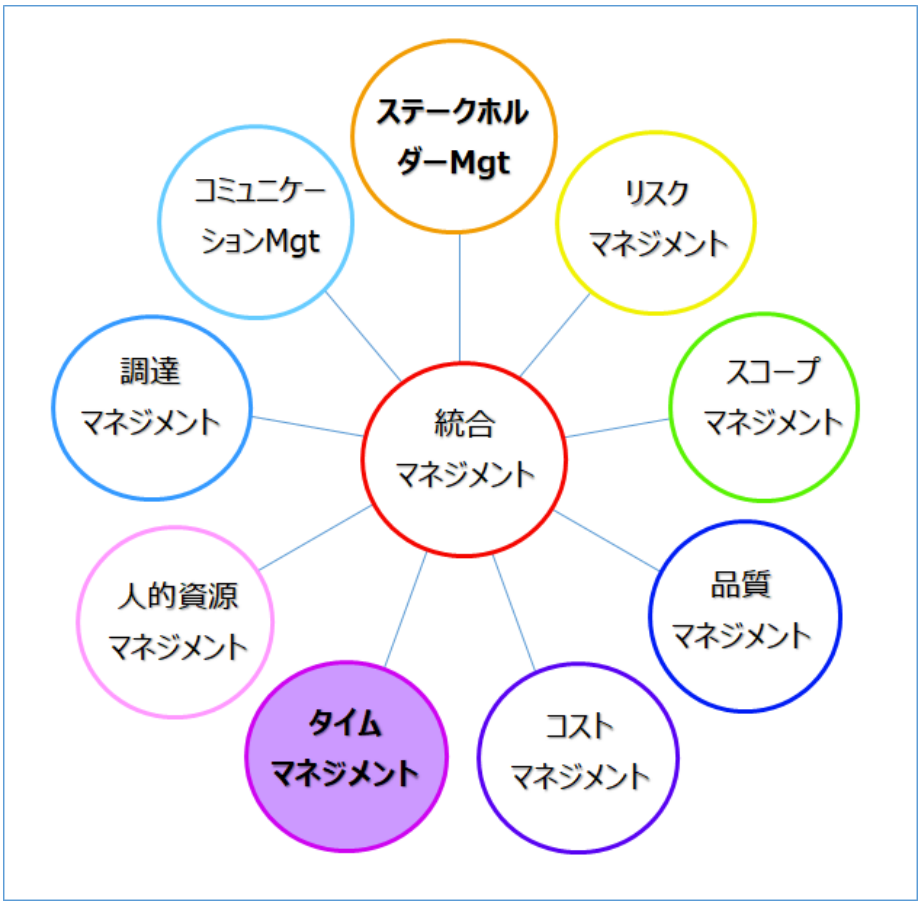
実用プロマネ　わかる現場の

タイムマネジメント



目次

管理工程　タイムマネジメント　ｐ６０

振り返り工程　タイムマネジメント　ｐ７３

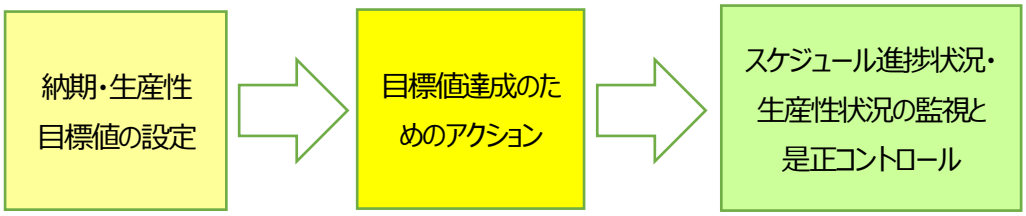
第１章　タイムマネジメントの役割

１－１．タイムマネジメントの役割

◎タイムマネジメントの役割

スケジュール目標を達成するように進捗をマネジメントすること。すなわち達成すべき納期および生産性の目標値を設定し、その目標を達成するための改善活動を伴った開発業務を遂行すること。

【タイムマネジメント】（図１―１）



１－２．プロジェクトの成功要件

一般的にソフトウェア開発プロジェクトの成功要件は、次の三つの目標の達成だと言われています。

【プロジェクトの成功要件】

１．品質目標　Ｑ：Ｑｕａｌｉｔｙ

２．コスト目標　Ｃ：Ｃｏｓｔ　⇒利益目標

３．納期目標　Ｄ：Ｄｅｌｉｖｅｒｙ　⇒生産性[1]目標

コスト目標は同時に利益目標となり、また納期目標は同時に生産性目標となります。

これらのＱＣＤの目標は、三つが同時に達成されて初めて成功プロジェクトだと言えます。

例えば下記の三つのケースにおける目標の達成はほとんど意味がないということが分かります。

① 品質目標は達成したがコスト・納期目標は未達成の場合

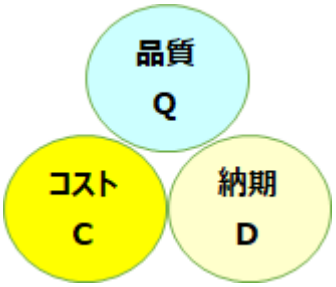
② コスト目標は達成したが品質・納期目標は未達成の場合

③ 納期目標は達成したが品質・コスト目標は未達成の場合

コスト目標は、開発組織内においては利益目標として表現されることもあります。

これらの三つの要件は顧客と開発組織との間の契約によって約束されるもので、どれか一つでも未達成ならば契約違反となり、民事的に公的な処罰の対象となることもあります。

【プロジェクト成功の３要件】（図１－１－３）



品質・コスト・納期目標の同時達成

[1] **生産性**　時間当りに生み出される成果物の量および質を表わす指標。

プロマネ

プロジェクトを成功に導くためにはプロマネは下記の実行が必要になります。

【プロジェクトを成功させるポイント】

① 妥当な見積りによる妥当な開発期間および開発工数の確保を行うこと。
② 要求仕様の早期凍結を行うこと。
③ リスクの早期解消を行うこと。
④ リーダーが開発仕様の全体像を押さえておくこと。
⑤ 開発メンバーに対して適切な仕事の配分を行い、十分に仕様の理解をさせておくこと。
⑥ 開発効率化・失敗防止のための継続的な改善活動[2]を実行すること。

効率的開発、良い品質、納期の確保、利益の創出を実現するためには上記の活動が必要になり、これらの活動を主導するのがプロマネの中心的な役割だと言えます。

PM

◎プロジェクトの成功要件は、目標ＱＣＤの三つを同時に達成すること。

[2] **改善活動** ＱＣＤに関する問題の改善を図りプロジェクトを成功に導くと同時に、人材の育成を図る活動のこと。特に失敗に学ぶ活動は有効な結果を生む。

第２章　事前準備工程　タイムマネジメント

２－１．顧客情報の収集

顧客が開発側に何を期待しているのかの情報

第一に入手が必要な情報は、顧客が開発側に何を期待しているのかを示す顧客特有の思考・行動様式に関する情報です。開発会社が接する顧客側の部署は、システム部とか情報企画部とか呼ばれる部署が一般的ですが、それらの部署はその会社特有の思考・行動様式を強く反映しています。プロジェクトを成功裏に終了するためには、彼らがどのような思考様式や行動様式をもっているのかを、開発が始まる前の受注活動の中で素早く理解しておく必要があります。

顧客側の思考・行動の代表的なパターンを示します。

身勝手な顧客の例

できるだけ安い金額と短納期でたくさんの仕様を盛り込み、良い品質のものを希望する顧客は、開発側にとっては困ったものですが、自分が車や家を購入する場合を振り返ってみると、一方的に非難することもできません。これが日本の消費者の典型的な、商品購入における思考・行動のパターンなのです。経験上このような顧客は全顧客のうちの約９割を占めていると思われます。 このような顧客に対して何の対策もなしに無防備に受注・開発を行うと以下のような問題が発生してくるでしょう。

①仕様の範囲（スコープ）がいつまでも確定されない

一旦決まった仕様ですら二転三転し、開発工程の後半になっても仕様変更や追加が止まらない。

②プロジェクトが危機に陥る

約束した開発期間や開発費は、開発半ばで消費し尽くしてしまう状況に陥る。

このような状況を回避するためには、事前準備工程の期間中に顧客の思考・行動に関する上記のリスクを排除しておく必要があります。下記のリスクヘッジ[3]策が有効です。

【身勝手な顧客に対する防衛策】

1．営業部署との連携による、顧客との密接なコミュニケーションを実行すること。
2．“新システム一式、仕様は打合せによる”などという基幹仕様未定な状態で受注はしないこと。
3．仕様凍結の最終期限を相互で約束しておくこと。
4．仕様凍結後の仕様変更・追加は別途見積り（開発費および開発期間）とすること。
5．短納期・低開発費を強く迫る顧客に対しては、開発済みの標準的な仕様の提案を行うこと。

身勝手な顧客に対する防衛策によって、顧客を妥当な要求に導き、開発側は妥当な開発期間・開発費を提示することで、両者ともに納得できる形で開発を進めることが可能になります。

さらにこれらのリスクヘッジは開発プロジェクトにおける最大のリスクである、要求仕様問題・見積り問題および開発行為のやり直しという三つのリスクの排除に大きく貢献します。

【プロジェクトにおける三大リスク】

1．いつまでも決まらない要求仕様

2．精度が低い見積り

3．開発行為のやり直し

[3] **リスクヘッジ**（Risk Hedge）損失・失敗の防衛策のこと。

開発側が知りたい情報の入手

開発側が最も知りたい情報は次の三つになります。

1．納期情報

2．要求仕様の骨子

3．発注予算情報

これらの情報は、事前準備の段階での入手が必要です。

あらかじめ顧客のこれらの情報を大まかにでも入手していれば、顧客との仕様検討や見積り等を開発側主導で進めることが可能になります。 これらの情報の入手は以下の行動によって可能になるでしょう。

1．営業部署と連携した、顧客との密接なコミュニケーションの実行による信頼関係の醸成。

2．失注競合他社の提案書などの情報入手。

準備

◎顧客の情報を知れば失敗リスクの半分は消える、
残りの半分はプロジェクト自身の開発力にある。

第３章　計画工程　タイムマネジメント

３－１．早期の仕様凍結

あいまいな要求仕様と無理な短納期･低コスト

プロジェクトの起点は、顧客あるいはベンダー[4]から提示される要求仕様および見積り依頼にあります。これはまた同時に、プロジェクトにおける問題の起点もここにあるということになります。開発の第一段階における要求仕様の品質および見積り回答の妥当性が、プロジェクトの成否を決定づけることになります。

顧客要求に関する主なリスクは次の通りです。

□ あいまいでいつまでも決まらない要求仕様

回避策 ⇒ 参照：ｐ１０◎仕様の理解

□ 無理な短納期･低コスト

回避策 ⇒ 参照：ｐ５【身勝手な顧客に対する防衛策】、
ｐ１８【見積りリスクの回避策】、ｐ５３●獲得時間の最大化

これらのリスクの解消法は、「仕様の早期凍結」および「妥当な納期･開発費の獲得」ですが、これを実現するためには、それ相応の仕様知識、技術力および交渉力が必要になり、その開発組織の能力レベルが上がるに従ってその成果も拡大していきます。

計画

◎プロジェクト問題の起点である要求仕様の品質は、ソフトウェアの品質と量を規定する。

【必要時間と許容時間】（図３－１－１）

[4] **ベンダー**（vendor）　ソフトウェアないしはハードウェア製品のメーカーまたは販売会社のこと。ユーザー企業から発注されるソフトウェア開発の元請け企業となる場合が多い。

計画

早期の仕様凍結

早期の仕様凍結のチェックポイントは次のようになります。

◆【早期仕様凍結のチェックポイント】

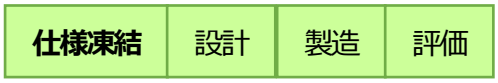
- □ 早期仕様凍結のために、顧客および関連各社の参加・協力の要請を行うこと。
- □ 仕様凍結の期限を切り、発注顧客側と受注開発側で合意をしておくこと。
- □ 顧客・開発会社間の直接コミュニケーションによる、要求仕様の期限内凍結を行うこと。
- □ 集中検討会ないしは合宿等にて、短期集中的に仕様決定を行うこと。
- □ 受注側においても提案型仕様凍結を行うこと。
- □ 顧客価値[5]の優先度順に、仕様凍結を行うこと。
- □ 顧客価値の高い仕様順に、開発着手すること。
- □ 開発仕様の目的・背景・範囲・内容を文書にて明確化すること。
- □ 基幹仕様未凍結状態ないしは疑問点・不明点を残したままで、先行開発着手は行わないこと。

計画

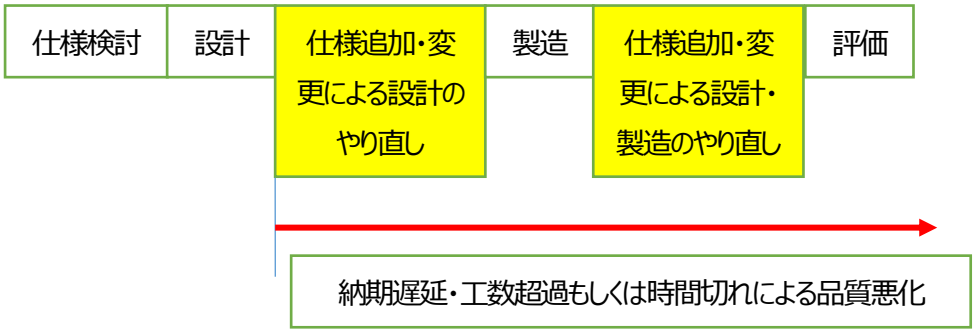
◎基本的な要求仕様は、設計着手前に凍結されなければならない。

【決まらない要求仕様がもたらす弊害】（図３－１－２）

<決まる仕様>



<決まらない仕様>



5 **顧客価値** 顧客がビジネス上重要視している機能やシステムのこと。

先行開発着手の得失について

要求仕様の決定が遅れ始めると、開発スケジュールを遅延させないために早期の仕様凍結をあきらめて先行着手[6]を始めるプロジェクトが後をたちません。それも開発の基幹仕様が決まっていないのにもかかわらず、先行着手に走ってしまっているのです。枝葉に相当するような仕様の先行開発ならば、後でやり直すとしても被害は微少で済みますが、基幹仕様の場合は、高い確率で大きなやり直しが発生します。
先行着手を行う開発者たちは、何をやるのかが決まってもいない仕様を、一体どうやって開発するのでしょうか。この場合、先行着手を別の言葉で言えば想定開発[7]ということになります。

早期の仕様凍結をあきらめた開発組織は、いつしか事前着手が当たり前の行為になってしまうだけではなく、二転三転し開発工程の後半になっても決まらない顧客の要求仕様に振り回されることになってしまいます。

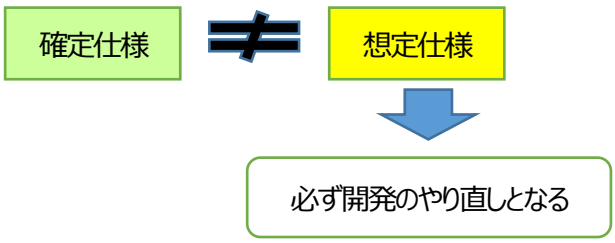
事前開発という想定開発によるやり直しに加えて、いつまでも決まらない要求仕様によるやり直しまでも背負うことになったプロジェクトは、必ずＱＣＤに大きな傷を負うことになります。先行着手で稼いだと思ったわずかな時間など何の効果もなかったことになります。

基幹仕様未凍結での開発着手を禁止するということは、非合理的な想定開発を禁止すると同時に早期の仕様凍結に全力で取り組まなければいけないということを意味しているのです。

この問題は、仕様決定の当事者になりにくい下請けのプロマネや開発リーダーをより深刻な状況に追い込むことになります。しかしこのような状態に追い込まれないためには、下請けのマネジメントは元請けのマネジメントに対して、顧客との仕様検討の場に下請けのプロマネや開発リーダーの参加を要請する必要があります。

早期仕様凍結を本当に実現したいと思うならば、元請け側がこの提案を断る理由はどこにもないでしょう。

【確定仕様と想定仕様】（図３－１－３）



---

6 **先行着手** 仕様未凍結状態で、開発側の想定仕様に基づいて開発を始めること。

7 **想定開発** 確定仕様に基づかず、自分の想定仕様による開発。多くの場合、やり直しが発生する。

計画

仕様の理解

優れた開発者における開発工程ごとの時間の使い方は、仕様の調査・検討である初期工程に重く、後になるに従って軽くなっています。さらに開発すべき対象をどれだけ正確に把握しているか、どれだけ早く理解しているか、がプロジェクトの成否の分かれ目になります。この勝負は見積り時および要件定義工程においてほぼ決定しています。プロジェクトを成功させる要因の重さは、"何を作るのか"が分かることで六割、"どのように作るのか"が分かることで三割、その他要因で一割程度だと思われます。

これらの事実を直感的に理解している開発者は、見積り時において、仕様の全体像の把握および主要な仕様の把握に全力を上げています。これができれば妥当な見積りが可能となり、基本設計の概要を描くことも可能になります。要するに、進むべき地図を先に明確に描いておくのか、それとも迷いながら進むべき道を探すのかの違いです。どちらが早く、しかも正確に目標に到達できるかは一目瞭然です。

計画

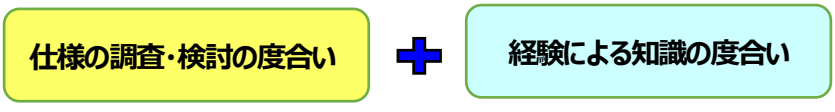
◎仕様の全体像および主要仕様の把握は、設計工程の前に済ませておくこと。

仕様凍結にあたってのチェックポイントは次のようになります。

◆【仕様凍結のチェックポイント】

- □ まずは仕様の全体像の把握から始めること。
- □ 要求仕様の背景や意味を必ず理解しておくこと。
- □ 仕様検討の段階で要求者と徹底的な仕様検討を行うこと。
- □ 疑問・不明点の発掘を行い、その解消に向けて、要求者に対し積極的な行動を取ること。
- □ 仕様決定のQ＆A[8]は直接対話による確認を行うこと。
- □ 不明なことは直ちに分かっている人・部署に聞くこと。
- □ 仕様検討にて新たに知り得た仕様や技術情報を、ドキュメント[9]によって他のメンバーに伝えること。
- □ 早期の仕様凍結を行うこと。
- □ 基幹仕様未決定で開発に着手しないこと。

【仕様の理解度】（図３－１－４）



[8] **Q＆A** Question＆Answer 質疑応答の略。ここでは仕様の不明点および疑問点に関する質疑応答のこと。

[9] **ドキュメント** 仕様書や設計書などに限らず管理用文書、ビジネス書類、メモなども含む全ての文書のこと。

計画

３－２．顧客要求の重要度順位の設定

基本的な仕様の確定およびＷＢＳの作成によって開発すべき顧客要求事項の内容はすべて明確に定義されました。続いて重要な作業として、プロジェクトの限られた時間と開発費を最大限有効に使うために、顧客要求事項の開発の優先順位を決めておく必要があります。

要求事項はすべて顧客の要求に基づいていますが、すべてが同じ顧客価値をもっているわけではありません。どの仕様がより顧客にとって重要かということは、要求仕様を決定する過程における顧客とのコミュニケーションの中で判断することができます。 重要度の設定作業は、まず複数の要求事項を顧客価値の重要度の順に、例えばＡ／Ｂ／Ｃの三つのグループに分けることから始めます。つまり開発の優先順位はＡ＞Ｂ＞Ｃのグループの順となります。 続いて各グループ内における優先順位の決め方として、Ａグループに属する仕様として例えばａ、ｂ、ｃが有った場合、ａを先に開発しなければｂもｃも動作できない場合は当然のことにａから着手することになります。このａをロードブロック[10]仕様（機能）と呼ぶことにします。同一の顧客価値内での優先順位はロードブロック仕様が優先権をもつことになります。ロードブロックの順に並べたものがプロセスの順になって来ます。 このように優先順位は、価値の順とプロセスの順という二つの意味合いを同時に判断する必要があります。

計画

◎優先順位の判断基準は次の二つ

１．顧客価値の重要度順

２．ロードブロック仕様の順

開発プロジェクトに起こりがちな優先順位無視の行動として以下のものがあり、プロマネは開発の全工程においてこれらの問題が発生しないように対策を行う必要があります。

【優先順位を無視した不正な開発行動】

- 仕様未決定状態における事前開発着手
- 要件定義書未完での基本設計の着手
- 基本設計未完での詳細設計の着手
- 詳細設計未完でのコーディング着手
- コーディング未完での評価テスト着手
- 総合評価未完でのソフトウェアのリリース[11]

---

10 **ロードブロック** 元の意味は道を塞いでいる障害物のこと。先に進むためにはその障害物を片付ける必要があると言うような文脈で使用される言葉。

11 **リリース** 完成したソフトウェアおよび関連成果物を顧客へ引き渡すこと。

これらの優先順位を無視した行動の主な原因は、開発時間および予算の不足にあります。時間が不足した結果、あせって事前着手に走り、その結果多くの手戻り作業を発生させ、また複数の仕事を同時に処理できず不十分な作業内容による全体的な品質低下を招くことになります。

妥当な優先順位を決定するためには、その前に見積り交渉などにおいて妥当な開発期間および開発費の獲得が必須条件となります。

次に開発作業におけるものごとの優先順位の決め方を下記に示します。

【開発作業における優先順位の決め方】

- 仕事の意味・意図・背景を仕事の着手前に把握すること。
- 仕事内容をブレークダウンすること。
- ブレークダウンした仕事内容に優先順位を設定すること。
- タイムリミットが直近に迫っているものから処理をすること（緊急度優先）。
- 時間的余裕があるものは重要機能の順に処理すること。
- 同等の優先順位のものは、効果が高く実行が容易なものの順で、効果が低く難易なものは最後に。

＊ 優先順位が判断できない場合は早めに顧客や上長に相談すること。

＊ この仕事に許される自分の残り時間を毎日意識すること。

作業の優先順位の決定にあたっては、次の三つの視点での検討が必要になります。

【優先順位決定に必要な３つの視点】

１．一つの作業内における処理の順番の妥当性の視点

２．複数の作業の緊急度・重要度の優先順位判断の視点

３．手持ち時間と必要時間の比較判断の視点

この三つの視点における判断ミスおよび必要時間の確保の失敗は、ＱＣＤを大きく毀損する結果を招くことになります。

また、優先順位の誤った認識としては次のようなものがあります。

【開発作業における優先順位を誤らせる原因】

- 割り込み作業等の優先度判断の誤り
- 必須業務に優先順位はないという認識の欠如
- 納期第一優先の誤り（ＱＣＤはみな等しい優先価値をもっている）

計画

【優先順位の原則】

顧客の要求は機能単位に分割され、顧客価値の優先順位の高い順に開発される。

３－３．見積り

見積りは、要件定義およびチームの技術能力とならんでプロジェクトを成功に導く三大要因の一つです。逆の表現をすると、見積り・要件定義・チームの技術能力はプロジェクトを失敗させる三大リスクとなっています。

見積り金額と開発期間（着手可能時期、開発完了時期）は、先に述べたＷＢＳの最小レベルの部材に要する金額と期間の総和として算出されることになりますが、見積り回答書を作成するにあたってはさまざまなリスクを考慮する必要があります。

ソフトウェアの価格

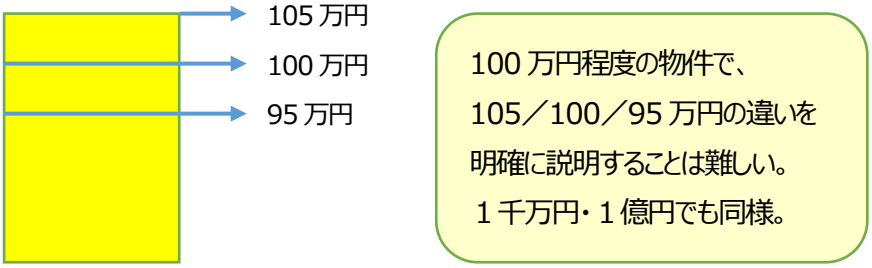
ソフトウエアの価格、特にカスタムソフト[12]の派生開発においては定価がないのが実状です。なぜなら顧客の要望に従って作成され、一本毎に内容が異なるため、いわゆる一品料理となり、全く同じものがこの世の中には存在しません。そのために価格についていくらが正当な価格なのか判断し難いのが実状です。

例えば１００万円程度の開発物件の場合を考えて見ると、この見積りにおいて１０５万円の場合、１００万円の場合、と９５万円の場合の違いを明白に区別できるでしょうか。 １０５万円と９５万円では実に１０ポイントもの差がでてしまいます。 優秀なプロマネならば発注者側の状況と自社側の状況を深く勘案しつつ、適正な価格の落とし所を見極めなければいけません。元値の１００万円という見積り内容にちゃんとした根拠があり、相手に納得させる能力さえあれば利益率の５％向上はそれほど難しいことではありません。

計画

◎有能なプロマネは＋５％の見積り額を獲得する。

【見積り価格】（図３－５－１）



[12] **カスタムソフト** 顧客からの特別な要求仕様に基づいて開発されるソフトウェア。対語は市販品であるパッケージソフトウェア。

計画

見積り回答書に対する認識

最初に見積り回答という行為に対する認識をしっかりと持つ必要があります。見積り回答書とは、法的な拘束力をもつ契約書です。受注者は発注者と約束したものを完成させる義務があり、発注者は受注者と約束した対価を支払う義務があります。この相互義務の履行を法的に約束した書類が見積り回答書です。見積り回答書は、単なるドキュメントとは訳が違うという認識が必要です。

計画

◎見積り回答書は、契約書である。

受注者にとっての見積り回答書の重要性

受注者の大きな目的の一つは、その仕事によって利益を上げることです。見積り回答書においては、達成すべき仕事内容、その対価および実行期間について約束してしまいますので、見積りミスや仕事の失敗は直ちに赤字を招き、企業の存続を危うくする場合もあります。

さらに見積り回答書は開発チームの仕事の原点であり、その後の開発における人・モノ・カネ・時間を規定するものであり、すべての書類やドキュメントの中でも最重要な書類です。見積り回答書は、その会社や組織のすべての実力を映し出す鏡といっても過言ではありません。

見積り回答書の基本的な要件は以下の二つだけです。

① 分かっている内容についてのみ見積ること
② 分かっていない内容については、見積りに含んでいないことを明記すること

すなわち見積り回答は、開発の対象となる機能について事前に過不足なく定義された要求書に基づいて行い、想像や想定に関することがらを一切含めてはいけません。

計画

◎見積りは分かっているものだけを見積ること。

計画

見積り回答書の品質

見積り回答書の形式的なチェック

妥当な見積り回答を行うためには、最初に見積りにおける記述モレ・考慮モレなどの単純なミスを防ぐために、見積り回答書の形式的なチェックが必要になります。モレなどのミスにつながりやすい項目としては、次のようなものがあります。

◆【見積り回答書の形式的なチェックリスト】

- □ 見積り回答期限の事前確認は行ったか。
- □ 見積り対象の仕様は明確になっているか。
- □ 見積りにインプット条件は全て網羅したか。
- □ 見積りにアウトプット条件は全て網羅したか。
- □ 仕様の疑問点・不明点は全て顧客に確認したか。
- □ 仕様変更が及ぼす影響範囲は特定したか。
- □ 特別な見積り条件の要求があった場合、その考慮は行ったか。
- □ 見積り範囲やリスクに関する条件を見積り回答書に記述したか。
- □ 見積り回答書に記述した開発範囲以外のものは別途見積りとするという文章を記述したか。
- □ 開発着手時期および完了時期を記入したか。
- □ 見積り有効期限を記入したか。
- □ ＊その他、過去の自他における見積り失敗項目を記述すること。

上記のチェックリストに、みなさんのプロジェクトで起こりがちなミスを加えれば、更に良いチェックリストになるでしょう。またこれらのミスを出さないような、見積り回答書の統一的なフォーマットやガイドラインの整備も必要となります。

計画

◎見積りガイドライン・見積りチェックリストによる単純ミスの防止を。

見積り精度の向上

次に必要なことは、見積りの内容自体の精度向上です。見積りの精度が低いために、妥当な開発費に対して異常に高い回答をすれば、顧客側の怒りを買い信用を失います。反対に誤って低い金額で回答してしまった場合、プロジェクトを成功裏に完了させることは不可能になってしまいます。開発期間についても同様のことが言えます。

一定の見積り精度を確保するための要点を次のチェックリストにまとめました。

◆【見積精度向上のチェックリスト】

- □ 見積り対象の仕様知識に習熟していること。
- □ 見積り対象システムのプログラム構造を理解していること。
- □ 見積り前に要求仕様の事前調査を済ませていること。
- □ 既存の設計書等が不備な場合、影響範囲に絞ったソースコードの調査を行うこと。
- □ 事前調査の結果をドキュメントに残しておくこと。
- □ 見積り対象仕様における、過去の開発の失敗事例を把握しておくこと。
- □ 見積り対象仕様の開発に必要な技術を保有していること。
- □ 過去の類似開発の見積り／実績データを参考にした見積りを行うこと。
- □ 見積り範囲（開発システムのスコープ）を明確にすること。
- □ 見積条件を明示すること。
- □ ソフト性能[13]設計の根拠となる、適用Ｈ／Ｗ等の性能値条件を明示すること。
- □ 要求仕様の精度レベルによって見積リスク係数を設定すること（非公開とする）。
- □ 流用部分はブロック図で表現すること（顧客に説明しやすい）。
- □ 添付資料の充実を図ること（運用フロー・データフロー等）。
- □ 納期によるステップ分けは別資料とすること。

大方の見積り方法は、経験に基づく手法で行われているようですが、単なる経験的な直感での見積りでは、失敗の危険性が非常に大きくなります。経験に基づくと言っても、何らかの経験知、すなわちデータに基づく必要があります。見積りに関する経験知データとしては、過去のプロジェクトにおける見積り値と実績値のデータがあります。毎回、見積りと実績の差異およびその原因・理由についての記録を取り、その改善対策を行うことで、チームの能力は強化され、次なる見積りの精度ははるかに高いものになります。全てのプロジェクトにおいて、見積りおよび実績値の記録および差異比較の振り返りは必須です。

---

13 **ソフト性能** ソフトウェアの動作速度、特に応答速度（レスポンス）および業務処理速度（パフォーマンス）は重要なソフトウェアの性能要件とされる。

計画

正確な見積りをするためには、まず仕様理解力と設計能力が必要です。見積りは基本的に、何をどう作るかが分かっていればできます。つまり明確な要件定義とそれを実現する設計が正しくできれば、見積りは可能です。要求仕様が明確なのに、正確な見積りができないということは、自分の仕様理解力と設計能力が未熟だということになります。見積り手法はあくまでも道具にすぎず、その手法があれば見積りができるというものではありません。ソロバンがあっても使えなければ意味がないのと同じです。まず自分の設計技術スキル向上と経験の積み重ねが必要です。

見積りの精度向上のためには上記の要件を満たす必要があり、当事者において能力不足の部分については他者の支援を仰ぐ必要があります。

計画

◎正確な見積りには、仕様理解力と設計能力が必要。

見積り方式

見積り手法の主なものには、①経験的手法 、②ＬＯＣ（Line of Code）[14]法 、
③ＦＰ（ファンクション・ポイント）[15]法などがあります。

①の経験的手法は広く行われている方法ですが、見積りの精度を上げるためには、過去のプロジェクトにおける、見積り対実績の差およびその原因についての分析データの蓄積が必要です。
②のＬＯＣ法は、工数＝ステップ数÷開発生産性ｘ（１＋間接費要員比率）ｘ余裕率で求められます。
③のＦＰ法は、工数＝基準値ｘ（０．６５＋調整値÷１００）で求められます。

②③の手法は、その開発組織としての今までの実績のデータの積み重ねがなければ、計算式のパラメータ値を決めることができません。 見積りの精度は見積り方式に依存するというよりも、見積り者が見積り対象の仕様ないしは機能に精通している度合いに依存しています。すなわち対象仕様・機能に関する開発経験および知識に決定的に依存しています。

計画

◎見積りの精度は見積り方式に依存するよりも、
仕様知識と開発能力に強く依存する。

---

[14] **ＬＯＣ**（Line of Code） ソフトウェアの規模を表す指標のひとつで、ソースコードの行数を意味する。

[15] **ＦＰ**（ファンクション・ポイント）ソフトウェアの機能数および複雑度による重みづけをした点数の合計から開発工数を見積る方式。

計画

見積りにおけるリスク

【リスクマネジメント】

見積りに関する主なリスクは次の通りです。

【見積りリスク】

① 要件定義書内容の不備（必要事項の記述なし・記述ミス、不明点・疑問点が多いなど）
② 要件定義書なしの見積り依頼（口頭・電話のみによる依頼など）
③ いつまでも決まらず二転三転する要求仕様
④ 短時間・短期間での見積り回答要求
⑤ 口頭や概算ベースの回答が正式回答として扱われる
⑥ 最初から仕切り値で開発費や納期を要求してくる場合
⑦ 特定の依頼者による、頻繁な受注につながらない見積り依頼
⑧ 見積り者における仕様の理解不足や事前調査の不足
⑨ 見積り者における交渉力の弱さ

見積り回答におけるリスク回避には次のような対策が必要になります。

【見積りリスクの回避対策】

① 見積り可能な要求仕様の提示がない見積り依頼は、出し直しを要求する。
② 妥当な見積り期間が設けられていない見積り要求には、必要期間を要求する。
③ 過剰な開発費削減・納期短縮要求に対しては、合理的根拠の提示を要求し安易な妥協はしない。
④ 口頭での見積り依頼に対しては口頭で返し、口頭レベルの依頼・回答は一切正式なものとしては扱わず、責任も持たない旨を通告しておく。
⑤ 公式のルートおよび承認のない見積り依頼は、出し直しを要求する。
⑥ 受注ヒット率が低い見積り依頼を頻繁に出す部署ないしは個人へ事情聴取を行う。
⑦ 短納期リスク物件においては、分割見積り・分割開発・分割リリース等の交渉を行う。
⑧ 複数社開発体制プロジェクトにおいては、自社責任範囲を見積り回答書に明記する。
⑨ 不慣れな仕様に関しては経験者の知見を借りる。

計画

◎リスキーな見積り依頼者が最大の見積りリスクである。
誰からの見積り依頼なのかに気をつけること。

見積りリスク回避対策の実行に当たって

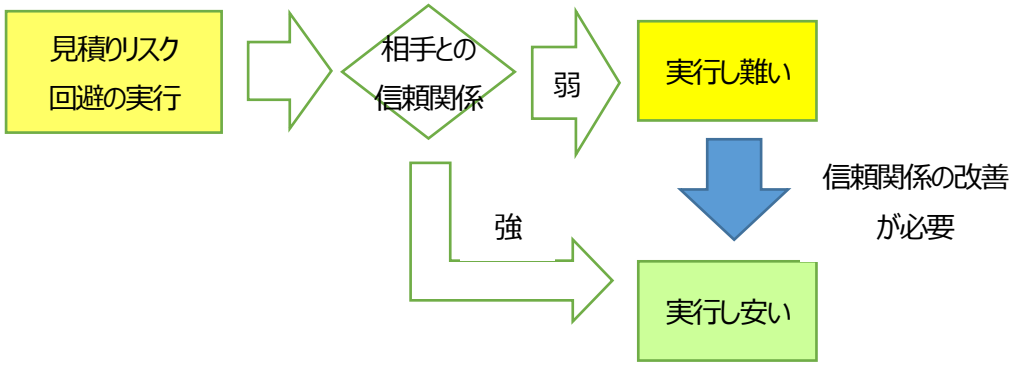
前記のリスク回避対策は、すぐに全てを実行することは無理なことでしょうが、これらのことは本来当たり前のことを当たり前に実行しようとしているだけのことです。

これらのリスク回避を行わなければ、まともなＱＣＤの達成はとてもできません。一つひとつの回避策の実行を重ねていくことが期待されます。良い製品を提供し、ＱＣＤを成立させるために必要ならば、一見無理だとか強硬に見えることに取り組まなければ、現状を打開することはできません。これらの施策を通すためには、まず良い品質の成果物を先に提供し続ける実績が必要で、良い製品を妥当な納期で出荷できた実績を積めば、見積り依頼部門も開発部門の要求に耳を傾けてくれるでしょう。

これらのリスク回避対策は、実際に開発組織において実行されたもので、架空の話ではないことを付け加えておきます。

そうは言っても、下請け開発会社が元請け開発会社に対して同様のリスク回避対策を行った場合、元請け担当者の怒りを買い、仕事を失うのではないのかという懸念の提起がありますが、元請けとの信頼関係が貧弱な場合はきっと怒りを買うことでしょう。そんな要求をする前に自分たちの開発品質をまともにしてください、と言われかねません。まずは自分たちの開発品質を合格レベルに向上させながら、これらの見積りリスク回避対策を順次実行していくことが順当なやり方でしょう。

【見積りリスク回避と相手との信頼関係】（図３－５－２）



概算見積り

正式見積りの前段階として、詳細仕様が未決定の状態で、概算見積りの要求を受ける場合がありますが、概算見積りは一見簡単なようですが、実はそう簡単なものではありません。概算＝いいかげん、では困ります。自分が精通している領域でなければ、概算見積りも詳細見積りもできないと言ってもよいでしょう。

調査をどの段階で見切るかと言う判断は難しいものだと思われます。自分における判断は、その見積り対象に対して、自分が保有する経験・知識および自分が利用できる他人の経験・知識の質と量によって決定されます。自分が利用できる経験・知識の質と量が大きい程、それによる判断は妥当性のあるものになります。

ある程度分かっているものならば、概算的見積りも可能でしょうが、利用できる経験・知識の総量（質）が少なければ、大雑把な見積りも困難になります。

細かく分析して工数や時間を見積もる詳細見積りは、いわゆる「定量的」見積りですが、大雑把に見当をつけるだけで良いのなら下記方法を試してみてください。

【概算見積りの方法】

① 要求仕様の全体像を、その目的・意味・背景を含めて把握すること。
② 自分で見積り可能なものとそうでないものを分ける。
③ 未経験項目および重要リスク項目をリストアップする。
④ それぞれに対して、必要工数を大雑把に３段階（大・中・小）または５段階に分けて記入する。
各段階に要する工数は、例えば、大＝１ヶ月、中＝２週間、小＝１週間等に決めておく。
重要リスクについても、もしそれが起きた場合についても同様に工数の大・小を記入しておく。
自分で決められないものについては経験者や上長の意見を聞く必要があります。
⑤ これらの仕事を何人で実行するかを想定しておく。
⑥ 非常に大雑把ですが、上記④の開発期間の合計を⑤の人数で割ったものが想定されるスケジュール期間になります。金額についても同様の方法で算出します。上記のやり方を、仕様検討工程・設計工程・製造工程・評価工程に分けて算出すれば、ある程度妥当な数字が出てくるでしょう。

そうは言っても、要求仕様の骨子も決まっていないような物件については、概算見積りすら不可能であり、このような場合には、要求者に対して早く要求仕様の骨子を決めるように促す必要があります。

計画

◎概算見積りはあくまでも参考値。
正式稟議承認には正式見積りを、の注意書きを忘れずに。

見積り回答書

一般的な見積り回答書に記載が必要な項目は次に示した通りです。

１．発行年月日
２．宛先、宛先（写し）
３．見積り回答書　：概算／正式　＊概算または正式を明示する。
４．承認印欄　：最終承認印欄・担当部長印欄・担当課長印欄
５．見積り依頼書番号欄、見積り回答書番号欄
６．開発名記述欄、ベース開発名記述欄
７．開発内容記述欄
　＊開発内容および範囲について漏れなく明示すること。
　＊見積りの根拠となる説明資料を添付すること。
８．開発条件記述欄
　＊ソフトウェア性能設計の根拠となる、適用ハードウェア等の性能条件を明示すること。
　＊要求内容不明等で見積りから除外した項目を明記しておくこと。
　＊開発内容記述欄に記述のないものは、別途見積りにする旨を明記すること。
９．開発費記述欄
　＊回答書表紙には開発費合計を記述し、主要項目ごとの明細開発費は添付資料とする。
１０．開発スケジュール記述欄
　□リードタイム[16]　□アクチュアル　＊リードタイムまたはアクチュアルスケジュールを明示する。
　＊アクチュアルスケジュールの回答は、着手可能年月日および開発完了年月日を明記する。
　＊リードタイムスケジュールの回答は、〇〇ヶ月の表記を行うこと。
１１．見積り担当者記述欄　：所属名・担当者氏名
１２．見積り有効期限年月日
１３．添付資料　□有：ｘｘ枚　□無

[16] **リードタイム**　開発に要する期間を示したもので、開発開始・終了等の時期は明示されていない。

計画

３－４．スケジュールの作成

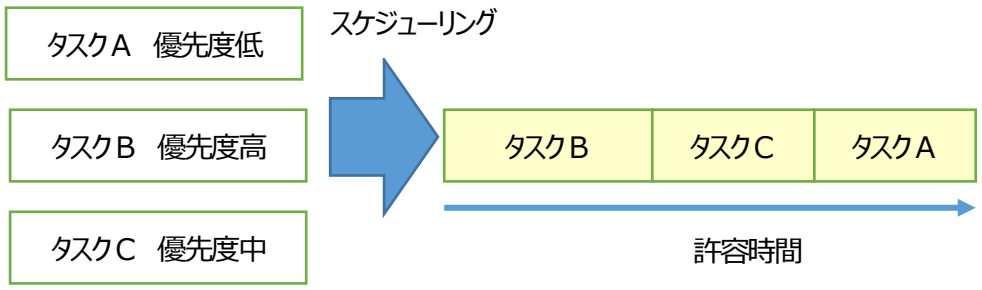
スケジュールの作成、すなわちスケジューリングとは、一言でいうと「持ち時間に仕事を割り振ること」です。やるべき仕事（タスク）をすべてブレークダウンし、それを持ち時間であるスケジュール表に投影させる事が、本来のスケジューリングです。 またスケジュール表は時間と仕事の相関関係を視覚化したものです。

スケジュールの調整が必要になった場合、先にスケジュール表の線引きをいじって、帳尻を合わせようとしてはいけません。 スケジュールの調整はまず仕事内容の見直しと、その裏付けである人員体制の調整から始めるべきで、その結果を新たに持ち時間に投影すべきです。また各タスク（仕事）間の関連性・継続性を意識することで、時系列的にバラバラになりがちなタスクを線として連続したものとして把握することも重要です。

計画

◎スケジュールの作成とはやるべき仕事（タスク）をすべてブレークダウンし、
それぞれをやるべき順番にプロジェクトの持ち時間に割り振って視覚化したもの。

【スケジューリング】（図３－６－１）



スケジュールの作成は、単純に言えばＷＢＳでブレークダウンした最小単位（レベル４）のタスクを時系列の順に並べることで作成されます。

例えば基本設計が、A・B・Cの三つのモジュールで構成されていた場合で、Aモジュールの設計に 2 週間、BおよびCは各１週間を要する場合の作業スケジュールは次のようになります。

【小日程表】（表３－６－１）

						作業日程		
項番	成果物	工数	開始日	終了日	担当者名	○月○日	△月△日	
1-1-1	Aモジュール設計書	2週間	○月○日	△月△日	a	→	→	
1-1-2	Bモジュール設計書	1週間	○月○日	△月△日	b	→		
1-1-3	Cモジュール設計書	1週間	○月○日	△月△日	c	→		

大日程および中日程表（ＷＢＳのレベル１～３）は、小日程表の積み上げ作業を行うことで作成されていきますが、顧客と合意済みの日程との食違いが発生しないように人員の投入数の調整や人材の入れ替えなどによる調整が必要になります。

一方、Aモジュールの設計が終わらなければBおよびCモジュールの設計に着手できない場合のスケジューリングは下記のようにすべきですが、日程が詰まっている場合に、何らの対策もないまま並列スケジューリングを無理やりしてしまうと、結局スケジュールは破綻してしまう結果となります。

【不適切なスケジューリング】（表３－６－２）

						作業日程		
項番	成果物	工数	開始日	終了日	担当者名	○月○日	△月△日	△月△日
1-1-1	Aモジュール設計書	2週間	○月○日	△月△日	a	→	→	
1-1-2	Bモジュール設計書	1週間	○月○日	△月△日	b	✖→		○→
1-1-3	Cモジュール設計書	1週間	○月○日	△月△日	c	✖→		○→

一般的な大日程は、調査、仕様検討、基本設計、詳細設計、製造、単体テスト、結合テスト、総合テストの各工程の順に、必要なタスクを先に着手すべきものから順に記述していきます。

計画

３－５．プロジェクト目標の設定

プロジェクト目標の代表的なものとして、ＱＣＤの目標すなわち品質目標、コスト･利益目標、納期･生産性目標の三つをあげることができます。

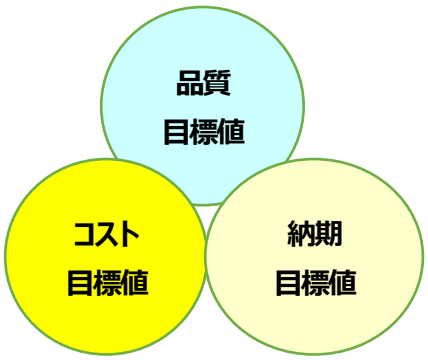
プロジェクトのＱＣＤ目標値は、見積り回答書にて示した数値内で設定することが絶対的な条件となります。次に、所属する上位組織のＱＣＤ目標に沿ったものが要求されますが、いずれにしても、その組織ないしはプロジェクトが持っている実力値以上の目標を達成することは極めて困難です。組織などの目標値の設定は、例えば三か年計画などに基づいて徐々に引き上げられていくことが現実的であり、達成の可能性も高くなります。 そのような意味で、プロジェクトの目標の設定にあたっては、最初に開発組織のＱＣＤの実力値を知るところから始める必要があります。

またＱＣＤの各目標は、同時に達成される必要があり、どれか一つないしは二つが達成されたとしても他の目標が達成されなかった場合は、成功プロジェクトとは言えません。

計画

◎ＱＣＤの目標値は顧客および社内経営層に約束したものであり、
どれか一つでも未達成ならば失敗プロジェクトとなる。

【ＱＣＤ目標値の設定】（図３－５－１）



スローガンではなく、目標数値の設定を！

計画

目標とは何か

目的[17]とは、「○○のために」という動機や理由を示すものです。一方、目標[18]とは、目的を達成するための具体的な手段を示すもので、「△△によって」というものです。例えば東京オリンピックについていえば、目的は「東京オリンピックを成功させるために」ということで、その目標は「何年何月までに１００億円を投じて５万名収容のスタジアムを完成させること」ということになるでしょう。「仕様理解力の向上」、「技術力の向上」、「評価業務の数値化」、「網羅性のある設計書の作成」、などという目標は、まだスローガン[19]レベルであり、具体性に乏しく、実現性の高い目標設定とはいえません。

われわれ実務レベルの開発者においては、自分の身近な仕事における、具体性のある問題を、数字をもって目標設定とする必要があります。例えば、仕様不具合を見抜けなかったことによって過去に発生した不具合の発生率・件数およびロス工数を具体的に数値で示し、それらの原因・真因を特定することで、具体的な対策案を提示し、改善効果として不具合件数を３０%削減する、ということを目標に設定するということです。

計画

◎目的とは、「○○のために」という動機や理由を示すもの。
目標とは、目的を達成するための具体的な手段を示すもの。

目標の見つけ方

より良い仕事をするためには、何が必要でしょうか。自分が把握できていない顧客の要望とは何でしょうか。コミュニケーションを図るとは、具体的にどのような手段で、何をするのでしょうか。無駄な作業にはどのようなものがありますか。本来やるべきことで、今までできなかったことにはどのようなものがありましたか。それぞれについて具体的な内容を書き出してみる必要があります。これらの具体的な内容から、自分の仕事における本当の目標が見つかります。まずは要求仕様の疑問点・不明点の解消および早期凍結に全力を注ぐ必要があります。さらに良い仕事を実現するためには、今までの失敗を振り返り、その改善行動を実行することが最も効果的な方法です。

計画

◎目標は、自他においての失敗や無駄の中にいくらでも見つけることができる。

---

17 **目的** ある行為を行う動機や理由を示したもの。「～のために」と表現されることが多い。

18 **目標** 目的を達成するための具体的な手段を示したもの。「～によって」と表現されることが多い。

19 **スローガン**（slogan） 目的や理念を表わしたイメージしやすい標語やキャッチコピーなど。

計画

目標の立て方

たとえば不具合発生件数０件ということは目標に相応しいと言えるでしょうか。不具合０件というのは一見、具体的でりっぱな目標のように見えますが、これはいわゆる「気合」や「意気込み」のたぐいのスローガンに過ぎません。１００ステップのプログラムでバグ０件はできるかも知れませんが、１万ステップのものではたぶん不可能でしょう。

「目標」と呼べるのにふさわしい条件は、まず現実的で達成可能であること、および測定可能な数値で示されることです。もし不具合件数を目標にするのなら、自分および他人における１ＫステップあたりのＸＸ不具合件数（単体テスト時、結合テスト時、総合テスト時）の過去の実績を記録に残しておき、目標値として自社の最高レベル～平均レベルの間の値を設定し、前期の自分の実績データと比較して今期は、３０％改善のＸＸ不具合件数目標＝○○件／１ＫステップなどをＸＸ設定すべきでしょう。これが合理的ないしは妥当性のある目標というものです。

計画

【目標の条件】

① 現実的で達成可能であること。

② 測定可能な数値で示されること。

【目標の立て方の順番】

① 現状の問題点を洗い出し、その数値化を行う。

② 問題点の真因を明確にし、改善策を立案する。

③ 改善策の期待効果に見合った目標値を設定する。

計画

納期・生産性目標値の設定

【タイムマネジメント】

納期の目標とは、単に納期の期日を守るということではないと言うことは誰でも分かっていることですが、切羽詰まって時間切れになってしまったプロジェクトが、多くのバグを内在したままでソフトウェアをリリースしてしまうことが少なからずあります。

納期が顧客にとって本当の価値を持つ大前提は、リリースされる製品が顧客の要求を満たしていると同時に品質が確保されていると言うことです。 言い換えれば、納期の目標とは、納期の期限内に品質を確保するための目標を設定することだと言えます。このことを実現するためには開発の各工程においても同様のことが実現される必要があります。すなわち次のようなことです。

- 要件定義の納期までに、品質が確保された要件定義を実行しなければならない。
- 基本設計の納期までに、品質が確保された基本設計を実行しなければならない。
- 詳細設計の納期までに、品質が確保された詳細設計を実行しなければならない。
- コード製造の納期までに、品質が確保されたコーディングを実行しなければならない。
- 総合テストの納期までに、品質が確保された総合テストを実行しなければならない。

計画

◎納期は品質が確保されなければ意味がない。

以上のことを実現するためには、それぞれの工程に許される期間内にそれぞれの成果物の品質を確保する必要があると言う当たり前の結論になりますが、問題のポイントは、許された時間よりそのプロジェクトが必要とする時間が長い場合にあります。必要時間[20]が長ければ当然時間不足となり品質の確保は不可能だと言えます。必要時間とは、言い換えればそのプロジェクトの実行能力（実力）を表わすものです。必要な時間を許容時間[21]内に収めるためにはプロジェクトの生産性能力を上げる以外には方法はありません。

計画

◎納期・品質の同時達成には生産性能力の向上が必要となる。

結論を言えば、納期目標の設定とは、要件定義／基本設計／詳細設計／コード製造／総合テストなどに関して期限までに品質を確保した成果物を OUTPUT するための生産性目標値の設定を行うと言うことになります。具体的には下記のような各成果物に対する品質レベルの目標値および時間あたりの生産効率が目標値となります。

---

20 **必要時間** ある仕事を実行するために必要な時間。実行する人や組織の能力により長短の差が出て来る。

21 **許容時間** ある仕事を実行するために許される一定の時間。

計画

各工程の納期が守られることは当然のこととして、それぞれの工程における生産性指標の例としては次のようなものがあります。

【生産性指標の例】

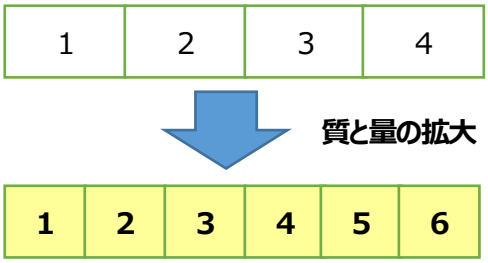
- 要件定義の生産性＝要件定義費用／要件定義時間
- 設計の生産性（設計効率）＝設計費用／設計時間
- 製造の生産性（製造効率）＝製造費用／製造時間
- 総合テストの生産性（テスト効率）＝総合テスト費用／総合テスト時間
- 納期達成率＝納期達成ＰＪ数／全ＰＪ数（ただし品質・コスト目標が達成されたＰＪのみ）
- 失敗による手戻り費用率＝手戻り費用／工程費用（開発各工程毎）
- 障害対応費用効率＝障害対応費用／全開発費

これらの指標値に基づいたプロジェクトの目標値の設定を行う必要があります。

計画

◎納期目標の設定は、成果物の精度目標および効率目標の設定に拠る。

【生産性：時間当たりの成果物】（図３－５－２）



納期・生産性に関する本章以降の解説の流れは下図のようになっています。

【計画のプロセス】
３－５．
●納期・生産性目標の設定 ⇒ｐ２７

【実行のプロセス】
４－２．
納期・生産性目標の達成
⇒ｐ３６

【管理のプロセス】
５－２．
進捗管理
⇒ｐ６４

計画

目標値設定による期待効果

【スコープマネジメント】

目標値の設定は、データに基づいた開発（データドリブン開発[22]）がプロジェクト遂行の第一歩と言え、ソフトウェア開発を科学的・工学的に遂行する基本的な手段となります。

数値に基づく開発は、プロジェクトの代表的な問題である、あいまいさや情緒的な判断を一掃する役割を果たします。

目標値の設定は次のような効果を生み出します。

【目標値設定による期待効果】

- 現状の開発力が数値として明確になる。
- 改善すべき問題を数値的に把握できる。
- 改善結果が数値的に明らかになる。
- 到達すべきゴールが数値的に見え、モチベーションの維持につながる。
- あいまいさや情緒性を排し、合理的ないしは妥当性のあるチームを育成する。

計画

◎目標は数値として表すことで、実行可能な目標となる。

[22] **データドリブン開発** データを根拠にした開発活動のこと。⇒参照：４－１．ＱＣＤ目標値の達成☆データドリブン開発とは何か

計画

目標をあきらめない

【スコープマネジメント】

仕事の目標は品質・コスト・納期の達成であると言われているのにもかかわらず、現実はコスト・納期の限界で品質を犠牲にしている場合が少なからずあります。コスト・納期の限界で多くの開発工程が時間切れとなり、それぞれの仕事が中途半端な未完成の状態で次の工程に回され、最後に市場において多くの障害を生んでいるのです。これは仕方ないといってあきらめても良い問題ではありません。中途半端な仕事でコストや納期を守ったつもりでも、結果として障害対応によってさらに多くのコストや時間を浪費することになり、最も大切な顧客の信用を失う結果を招いてしまいます。

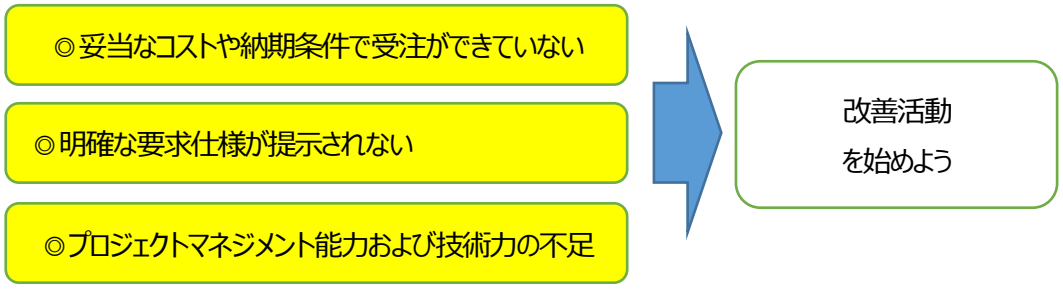
なぜコスト不足・時間不足になっているのか真剣に考える必要があります。多くの問題は、要件定義や見積りなど開発の前工程に存在していることは誰でも気づいていることですが、多くの人はその改善に着手しようとはしません。改善活動は仕事には含まれていないと考えている開発者やマネージャが意外に多いのです。

改善活動こそ仕事の中核業務であるという意識を持つ必要があります。

失敗プロジェクトの一番の問題は、妥当なコストや納期条件での受注ができていないこと、二番目の問題は明確な要求仕様が提示されないこと、三番目の問題は合理的なプロジェクトマネジメント能力および技術力の不足にあります。この問題を解決するためには現在までの自分たちの失敗を直視し、その真因を把握し、その改善目標を設定し、改善活動を実行・継続する必要があります。

目標の達成をあきらめたらその時点で失敗が確定してしまうことになります。

【失敗プロジェクトの原因ワースト３】（図３－５－３）



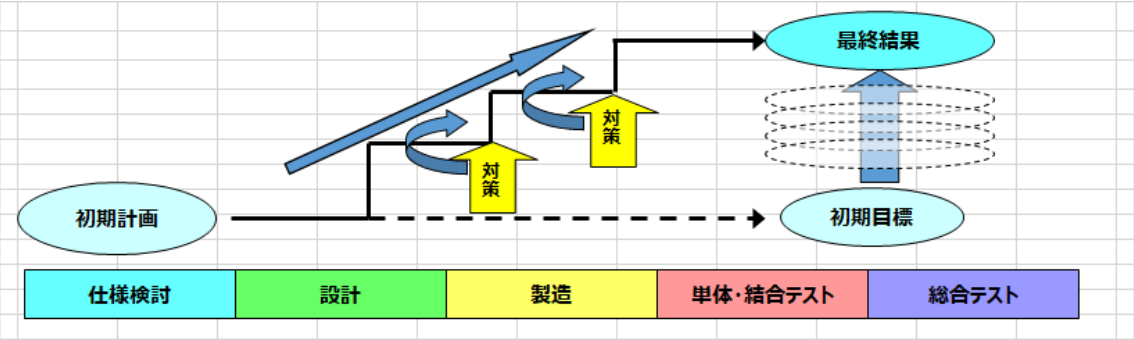
計画

◎目標は、あきらめなければ必ず達成可能である。

【目標は動いている】

到達目標は、初期に想定した位置から時間の経過・環境・条件の変化に伴い必ず変化していくものです。動的に動く目標をヒットするためには、変化を見逃さずタイミングの良い対策を連続して実施することが必要です。この事を図で表すと次のようになります。

【目標は動いている】（図３－５－４）



プロマネは、開発の進行とともに変化していく顧客の要求や開発の状況を見誤ることなく、それぞれの問題に対して適切な対策を適切なタイミングで打ち続けることで、プロジェクトの成功という最終目標を達成する必要があります。

【計画工程のまとめ】

本章の解説の流れをまとめると以下のようになります。

計画

【タイムマネジメントに関する開発条件の計画化】

１．早期の仕様凍結	：何を開発すべきかを設計着手前に明確に決定する。
２．顧客要求の重要度順位の設定	：要求仕様の開発優先度を明確にする。
３．見積り	：ＷＢＳに基づいて工数および開発期間を算出する。
４．スケジュールの作成	：ＷＢＳおよび優先度に従った開発スケジュールの作成。
５．プロジェクト目標の設定	：品質･コスト･納期（生産性）の目標を設定する。

６．プロジェクト計画書の作成

上記のすべての情報を集約し、開発実行の基本計画書を作成する。

計画の工程は、プロジェクトが具体的な開発行為の着手を可能にするために必要なすべての条件を整えるプロセスだと言えます。 タイムマネジメントの視点から見た場合、１．から５．までは開発行為の優先順位に並べられた開発着手を可能にするための条件であり、６．で作成される「プロジェクト計画書」はこれらの全ての条件を集約したものです。

プロジェクト計画書は次の「実行工程」における基本的な行動計画書となります。

第４章　実行工程　タイムマネジメント

４－１．ＱＣＤ目標値の達成☆データドリブン開発の実行

データドリブン開発とは何か

データドリブン開発とは、開発の推進にあたってデータを根拠にした活動を推し進めるという考え方です。

データを根拠にするとは、事実を証明する数値に基づいた資料や論拠を使うと言うことです。具体的には開発における目標の設定、問題の分析、および開発行為のすべてに必要なコミュニケーションや、要件定義書・設計書などのドキュメントの記述内容は、全て数値・数字の裏付けが必要だと言うことです。

難しく複雑な開発という仕事を無事に完了させるためには、感情論を排し、合理的な方法が必要になります。合理的な仕事のやり方の基本が、データに基づくやり方およびドキュメントに基づいたやり方です。数値で示されるものでなければ、目標も結果も適切に判断することはできません。

データに基づいて開発を遂行するということは、自分ないしは自チームの過去の QCD 等のデータおよびそれらから導き出された目標値データに基づいて仕事をするということです。データに基づかない仕事のやり方は、出たとこ勝負のやり方でプロの仕事とは言えません。

実行

◎基本的なデータはＱＣＤのデータ。

実行

データドリブン開発の基本的なプロセス

開発の基本的な数値はＱＣＤに関する数値です。開発チームにおける現在のＱＣＤの実力値を知り、目標とするＱＣＤの値を設定し、プロジェクト終了時に達成されたＱＣＤの数値の振り返りを行うことがデータドリブン開発の基本的なプロセスだと言えます。

プロジェクトを開始する前に、過去のプロジェクトにおける品質に関するデータを収集しておく必要があります。この過去のデータの平均値が現時点における開発チームの実力値を表わしています。これまでの失敗の反省に基づき、改善対策を実行した結果、得られる期待成果[23]としての数値がプロジェクトの目標値となります。この目標値はプロジェクトの終了時に得られた実績数値との比較を行うことで、次のプロジェクトにおける目標指標を算出することができます。

ＱＣＤの目標値の設定および具体的な指標例については、「３－５．プロジェクト目標の設定」において触れた通りですが、本章においてはそれらの目標値を達成するために必要な活動について解説をします。

実行

◎過去のデータ、目標とするデータ、結果のデータの三つでドライブする。

データドリブン開発を構成する三つの要素

データに基づく開発は、次の三つの開発コンセプト[24]において推進されることで、ＱＣＤ目標値の実現を確かなものにします。

１．ドキュメントベース開発[25]　　Ｑ：品質目標値の達成
２．プロフィットドリブン開発[26]　　Ｃ：利益･コスト目標値の達成
３．優先順位ベース開発[27]　　Ｄ：納期･生産性目標値の達成

それぞれの開発コンセプトの実行はＱＣＤのすべてに好影響を与えますが、上記はＱＣＤのどれに強力に効果をあらわすのかということを表わしたものです。

---

23 **期待成果**　達成したいと考えている目標数値。
24 **コンセプト**　ものごとの基本的な概念を言い表したもの。ものごとの骨格を表わす考え方。
25 **ドキュメントベース開発**　口頭だけに依存せず、書類･書面に基づいて実行される開発。
26 **プロフィットドリブン開発**　利益目標の達成を主眼として、同時にコスト目標の達成を実現する開発。
27 **優先順位ベース開発**　顧客価値の高い要求仕様の開発を優先させ、不要不急な開発を避けること。

データドリブン開発の効用

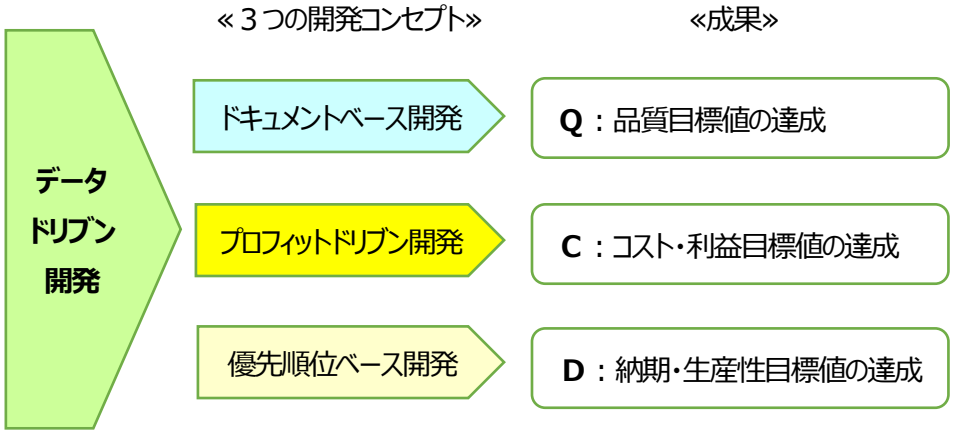
データに基づいた開発をすると言うことは、最短時間・最小エネルギーで物事を成し遂げる最善の方法として、合理的なやり方を採用するということです。合理性とは、物事の筋道を数値によって明らかにし、その数値の意味するところによって現状を把握し、それに従った行動を行うということです。日常業務の多くの物事は数値によって表現が可能であり、合理性の効用として以下のものが考えられます。

【合理性の効用】

① 種々に異なる問題の解釈を、だれの誤解も招かない一つの形式で表すことができる。
② 問題の量や質など、度量衡に関する全てのことを具体的かつ明確に示すことができる。
③ 数値化された原因や結果は、衆人の検証に耐えうる証拠となり得る。
④ 数値化することで見えない事象、すなわち見えない原因や結果を見えるようにできる。
⑤ 最短の時間と最小のエネルギーで問題を解決できる。

多くの問題は、物事を数値・数理で表すことによって解決可能です。特に開発上の問題は、品質・コスト・時間に関するデータや数値の把握によらなければ絶対に解決することはできません。数値でとらえるべき問題を情緒的な判断でとらえようとしても適切な解は得られず、時間とエネルギーの無駄使いとなるだけです。

【データドリブン開発】（図４－１－１）



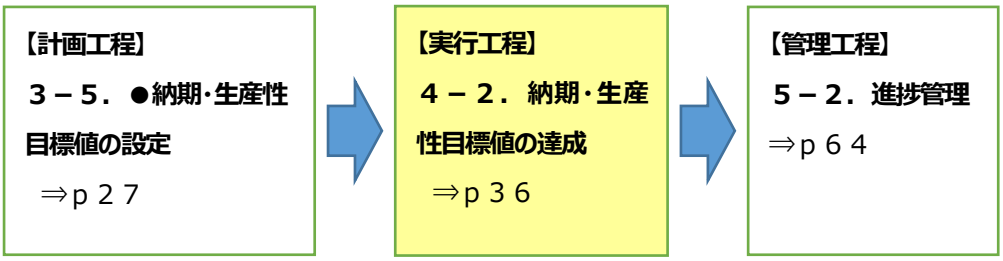
４－２．納期･生産性目標値の達成☆優先順位ベース開発の実行

一般的な物事の実行において、その着手する優先順位は非常に重要なことです。瑣末な問題に多くの時間を費やしていては、本当に重要な問題の解決に遅れをとってしまい、計画自体が中途で頓挫してしまうことがよくあるものです。

納期･生産性目標値の達成は、主に優先順位ベース開発の実行および後述の「４－３．改善活動の実行」によって実現されます。

納期･生産性目標値の達成

計画工程の３－５．において納期･生産性目標値の設定について解説を行いましたが、本章においてはその目標値達成のために必要なプラクティスについての解説を行います。納期･生産性に関する解説の流れは下図のようになっています。



納期そのものは名目上の目標に過ぎず、納期は製品の品質やコストの目標が達成されて初めて意味を持つ指標となります。不具合が多発するような製品を納期にリリースすることなど納期を守ったなどとは言えません。

意味のある納期を支える指標としては、時間効率に関する生産性の指標が意味のある指標になります。

納期･生産性の目標値の設定については、第３章　計画工程の３－５．において説明した通りですが、これらの生産性目標値を達成するためには、本章で示しているすべてのプラクティスの実行が必要になります。

◎納期の達成とは、
見かけ上の納期を守ることではなく、
品質目標･コスト（利益）目標も含めて同時に達成すること。

実行

第３章の３－５．で示した生産性指標を再度示しておきます。

【生産性指標】

- 要件定義の生産性＝要件定義費用／要件定義時間
- 設計の生産性（設計効率）＝設計費用／設計時間
- 製造の生産性（製造効率）＝製造費用／製造時間
- 総合テストの生産性（テスト効率）＝総合テスト費用／総合テスト時間
- 納期達成率＝納期達成ＰＪ数／全ＰＪ数（ただし品質･コスト目標が達成されたＰＪのみ）
- 失敗による手戻り費用率＝手戻り費用／工程費用（開発各工程毎）
- 障害対応費用効率＝障害対応費用／全開発費

実行

◎すべての生産性指標は改善活動のテーマであり、

組織的な改善活動を実行しなければ上がることはない。

開発業務における実行の優先順位

一般的なものごとの実行の優先順位は次の通りです。

１．顧客価値の高い順。

２．次に、効果が大きく実行が容易なものの順。

３．タイムリミットが必須のものはその期限までに実行しなければならない。

優先順位に従った開発業務の実行手順は次のようになります。

要求仕様を顧客価値の重要度順に整理すること

顧客価値の重要度の順位は、顧客との密接なコミュニケーションを図ることで、概要要求仕様の検討段階で判断可能です。発注側･受注側において仕様の概要が分かった時点で、仕様開発の優先度の順位を決定･合意することが重要です。優先度の順位は、リリース時期の先･後を考慮に入れた上での顧客価値の重要度の順とすべきでしょう。

実行

◎無駄な開発の排除のために

① 顧客価値のない要件を排除すること。

② 顧客価値の低い要件の優先順位を下げること。

顧客価値の重要度順に仕様を凍結していくこと

仕様凍結にあたっては、仕様を全部一時に決めようとしないで、重要度順に凍結していくことが必要であり、更に顧客価値を満足させる要件定義の実行、すなわち開発目標の明確化が必要です。これらは要件定義にあたって、顧客と開発チームの密接なコミュニケーションをベースに、顧客の要求について顧客価値の優先順位付けを行い、優先度の高いものを顧客の必要とする時期に提供していく計画を立て、さらには顧客価値の低い開発行為の優先順位を下げることにより、顧客にとっても必要性の低い開発行為を排除すること、などを通して顧客の要求と開発行為を一致させることによって実現されます。

これらの要件定義における基本的な取り組みの実行は、要件およびその範囲（スコープ）の明確化をメリハリのあるものにし、その変更管理のコントロールを容易にします。さらに要件の仕様化にあたっては仕様の階層構造的記述（ＷＢＳ）が、仕様モレ防止・解釈違い防止・矛盾仕様の発見・正確性・分かりやすさに寄与するでしょう。

実行

◎有効な開発に要するエネルギーの最小化を図るために

① 顧客の要求と開発行為を一致させること。

② 顧客価値の優先度の順・必要とされる時期に成果物を提供していくこと。

集中仕様凍結会議の実施

仕様検討に時間がかかりそうな場合は、日時を決めた上で顧客と直接顔を付き合せ、こちらからも対案を出し、検討会や合宿で一気に仕様決定を行う必要があります。相手が決めてくれるのを待っているような受け身の姿勢では何も改善できません。

顧客価値の重要度順に凍結した仕様から開発着手を行うこと

顧客との間で合意された優先度の順位に従って、詳細仕様の決定、ＷＢＳの作成に基づいたスケジューリングを行い開発に着手します。 顧客との間で合意された開発の実行順位に関する情報は、逐次プロジェクトチーム内の全員に共有される必要があります。

【顧客価値の高い順の開発実行】（図４－２－１）

実行

ＱＣＤには優先順位はつけられない

プロジェクトの成功と失敗について、日経ＢＰ社による２００９年の「プロジェクト実態調査８００社」によればプロジェクトの成功率は３１．１%だったという分析結果が出ています。
（Q　品質は成功：51.9%、C　コストは成功：63.2%、D　納期は成功：54.6%）

この調査結果から見えることは次の通りです。
開発会社は表向き品質や納期優先だとは言っていますが、多くの失敗プロジェクトにおけるＱＣＤの結果を見る限りにおいて、その実態はコスト、すなわち利益第一優先で、見かけ上納期を守ったように装い、品質を犠牲にしている姿が見えてきます。
すなわち、現実のＱＣＤの優先順位はコスト（利益）＞納期＞品質の順になっています。
これで良いのでしょうか？これで良いわけがありません、顧客は怒ります。

会社を潰さないようにする視点で見れば、コスト優先になるのでしょう。コストはマネージャ以外には見えにくい指標ですから、一般の開発者に見える部分では納期優先となってしまいます。約束の時期を外すことはできないために、最後の評価・デバッグの時間が不十分だったとしても、タイムアウトということになり、バグ含みで出荷されるソフトウェアが後を絶ちません。コストを守るために、必要な開発時間をカットして納期の帳尻を合わせ、そのために品質を犠牲にしているという図式です。もしこの通りの優先順位で実行したら、顧客の満足や品質第一が優先だと言っていることは嘘になるでしょう。悪い品質のものを買わされて被害を受けるのは顧客自身です。

品質・コスト・納期に優先順位はなく、本来同列の価値を持つものです。皆必須条件なのです。どれかだけを優先し他を犠牲にすることはできません。人間において、脳を取りますか心臓を取りますか選んで下さいというのと同じです。

やるべきことは、その様な究極の選択をしなければならないような状況に陥らないようにすることです。失敗するプロジェクトの主な原因は、仕様凍結遅れ・開発プロセスの悪さ・リスク回避の失敗などにあります。つまり「何をつくるか」「どうつくるか」が不明確なため、大きな時間ロスを生んでいるということです。現実的に色々な悪条件が存在するプロジェクトにおいては、上記のことを十分に承知した上で、妥当な品質・妥当なコスト・妥当な納期を追及することがポイントになります。

実行

◎ＱＣＤに優先順位を設けてはいけない

実態は

コスト（利益）＞納期＞品質

実行

４－３．改善活動の実行

改善活動と言えば、大方の人たちは時間の余裕がある時に仕事のついでに行うＱＣサークル活動のようなものだと思っています。 時間がないから、予算がないから改善活動ができないという声をあちこちで聞きます。では、時間やお金があったら本当に改善活動ができるのでしょうか。そのような人たちに時間と金を与えても、結局形ばかりの活動になり、何らの実利も得られないでしょう。成長の意欲のある人は、他人から言われずとも知恵を出して、時間を生み出し、利益を生み出すための改善活動をすでに実行しています。成長意欲のない人には改善の意義も理解できず、改善の動機も湧かないのでしょう。

そういうわけで、時間がない、資金もない、要求仕様も二転三転して定まらないというような、愚痴ばかりを並べたてるばかりで、一向にそれらの改善活動も実行せず、仕事の内容は手抜きや丸投げが蔓延するばかりで、製品は市場でバグだらけで、損益も万年赤字続きが続いているのです。

改善活動の意義

このような荒廃した環境を作るのも復興させるのも人間、とくに組織をリードする人たちです。プロマネの役割と責任を自覚する必要があるでしょう。個人および組織に、実利の獲得と人間的成長を同時にもたらす運動の一丁目一番地は、コミュニケーションの活性化を基盤とした改善活動に他なりません。改善活動は開発業務の中核業務であり、暇な時についでにやるようなものではないのです。

改善活動がもたらす成果には次のようなものがあります。

【改善活動がもたらす実り】

１．ＱＣＤ数値の大幅な改善
２．チームメンバーの技術能力の向上
３．チームリーダーのプロマネ能力の向上
４．コミュニケーションの活性化による組織能力の向上
５．全員の自律（自立）能力およびモチベーションの向上
６．会社の成長・発展

改善活動の苦痛を嫌って現在の地獄的な状況を我慢し続けるのか、一時の新たな負担を背負う覚悟を決めて改善活動の実りを手に入れるのかを決めるのは、みなさんの選択次第なのです。

実行

◎改善活動は、開発業務の中核業務であり、暇な時についでにやるようなものではない。

仕事に対する認識

ルーチンワークをこなしているだけでは、品質の高い業務や製品を生み出すことはできません。ルーチンワークと合わせて、自他の過去の失敗に学ぶ改善活動の実行が、高品質な業務および製品を生み出すことはトヨタのカイゼン活動で実証済みであり、世界中に知られていることです。

改善活動を行っていない開発組織は、開発という創造的な仕事から、いつの間にかルーチンワークという単純労働集約型の仕事へと劣化していきます。良い仕事を継続的に実行していくためには、改善活動を中核にすえた仕事を日常的に実行する必要があります。

実行

◎**良い仕事＝改善活動＋ルーチンワーク**

開発業務＝創造的仕事＋ルーチンワーク

改善活動の基本コンセプト

改善活動の基本コンセプトは以下の通りです。

【改善活動の基本コンセプト】

① 個人戦[28]ではなく組織戦[29]（チームプレー）を行うこと。
② 一人だけが成長するのではなく、みな共に成長すること。
③ 能力に長けたものは、後進の者にその知恵（ノウハウ）を譲ること。
④ 後進の者もその成長に従って、さらに後に続く後進の者に、その知恵（ノウハウ）を譲ること。
⑤ 先に進んでいる者は、その持てるあらゆる価値あるもの、資産・資金・知恵（ノウハウ）の全てを後進のものに順に譲り渡し、永続的な繁栄の循環を実現させること。

実行

◎**改善活動の基本コンセプト**

共に成長し、持てる智・財を継承し、永続的な繁栄の循環を実現すること。

---

[28] **個人戦** 組織に拠ることなく、個人単独で問題解決に当たること。

[29] **組織戦** 複数の人間で組織を形成し、それぞれに役割分担を決め、協調・連携のもとに問題解決に当たること。

改善活動の事例

改善活動の具体的な例としては以下のものがあります。

【改善活動の事例】

１．コミュニケーションの活性化
２．開発プロセスの確立と励行
３．プロジェクトの数値目標設定
４．要求仕様書の精度向上
５．要求仕様の早期凍結化
６．要求仕様の変更管理
７．仕様変更影響度表の作成
８．見積り手法問題の改善
９．見積りリスクの排除
１０．開発リスクの排除
１１．設計書の精度向上
１２．評価仕様書の精度向上
１３．無駄の排除
１４．開発の効率化
１５．レビュー品質問題の改善
１６．プロジェクトの振り返り（ラップアップ）励行
１７．開発ノウハウの継承
１８．リーダーシップとチームプレー問題の改善
１９．モチベーションの喚起
２０．手抜き問題の撲滅
２１．組織風土文化の改善

いずれも身近にある問題ばかりです。これらの問題を全てとは言わないまでも１０%、いや３０%程度改善するだけで、どれ程の成果がプロジェクトのみならず会社にもたらされるのかは容易に想像できるでしょう。

◎改善活動の具体的なテーマは、
繰り返される失敗や、いつもリスク管理表に取り上げられる項目の中にある。

改善活動計画書

改善活動計画書の基本的な記述内容を下記に示します。

この計画書はいわゆるＤＭＡＩＣ[30]手法に準じたもので、ＤＭＡＩＣの意味は次の通りです。

Ｄ（Define：問題点記述）

Ｍ（Measure：現状の指標値データ）

Ａ（Analyze：現状のデータ分析）

Ｉ（Improve：解決策）

Ｃ（Control：成果の維持・定着方法）

【改善活動計画書フォーム】

１．プロジェクトテーマ名

２．プロジェクト体制　：リーダー名記述、メンバー名記述

３．プロジェクト期間　：開始日、終了予定日

４．問題点記述（Ｄｅｆｉｎｅ）：＊解決すべき課題を現在の状況と共に複数記述する。

５．顧客に提供される価値（顧客のＣＴＱ　Critical　To　Quality）

＊この改善によって顧客に提供される価値（ＱＣＤ等）を記述する。

６．プロジェクトの成果目標

・目標の記述　＊何をどのようにするかを記述する。

・成果目標値の記述　＊改善活動によるＱＣＤ等の成果期待値・金額等を記述する。

７．現状の指標（値）データ（Ｍｅａｓｕｒｅ）

・収集した現状の問題等を整理・分類し、データ化する。改善目標について現状の数値を示す。

８．現状のデータ分析（Ａｎａｌｙｚｅ）

・フィッシュボーンによる分析；問題を構成している要因分析を行い、主要要因をあぶり出す。

・データに基づく分析；問題を構成している複数の要因を数値分析し、改善ターゲットを特定する。

９．解決策（Ｉｍｐｒｏｖｅ）

・複数の解決策を立案し、実行の優先順位をつける。

・実行の優先順位は、ＱＣＤ改善効果（費用対効果）＋実行難易度＋緊急度により決定する。

１０．成果の達成見込み（＊改善実行後には実績値に書き換える）

・成果物、改善金額、投資コスト、成果金額（＝改善金額－投資コスト）等。

１１．成果の維持・定着方法（Ｃｏｎｔｒｏｌ）

・実行された解決策が将来に渡って組織に定着する方法・手段を記述する。

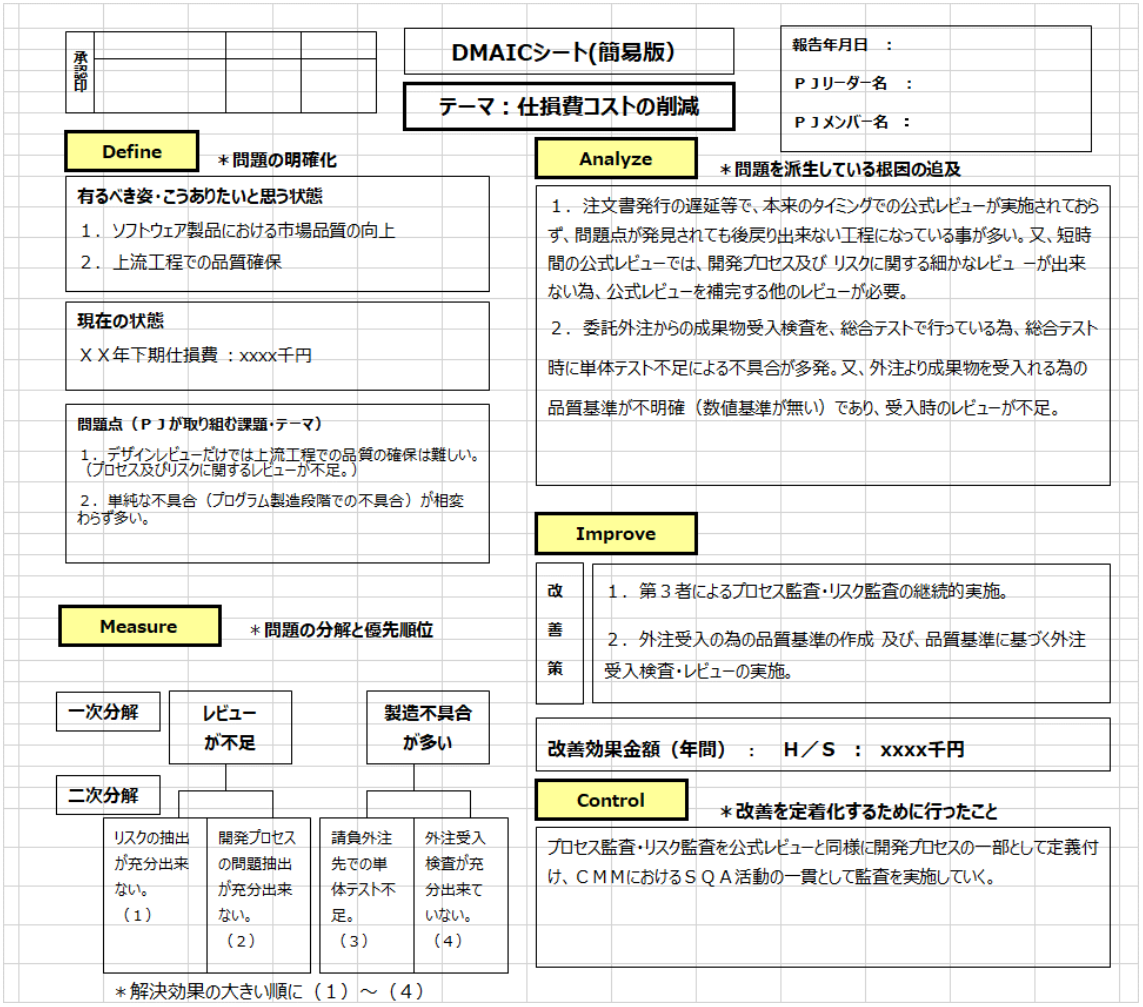
---

[30] **ＤＭＡＩＣ**　Define（定義）、Measure（測定）、Analyze（分析）、Improve（解決策）、Control（管理）のステップからなる経営変革手法であり、VOC（Voice of Customer、顧客の声）を基にして事業活動を分析し、データドリブンでプロセスの改善を進める。

実行

下記は改善計画書簡易版のサンプルです。

【ＤＭＡＩＣシート簡易版】（図４－３－１）



実行

改善活動を阻害するもの

改善活動を阻害する要因として、心理的なものと環境的なものの二つがあります。

心理的な要因

心理的な要因としては、たとえば改善活動によって将来大きな利益が得られると言われても、その失敗リスクを考えると現状のままの方がまだ安全だという心理が働き、たとえ、品質が悪い・コストが高い・生産性が悪いという状況であったとしても、何とかやっていくことが出来ているという、低レベルではあるが、それなりに安定している現状を失いたくないという心理が働きます。要するに、悪いなりに安定している現状を失いたくないために、将来利益を生み出すかも知れない改善活動を避けようとしているものと考えられます。また現状のＱＣＤのレベルはたとえ余り良くないとは思っていても、顧客からの苦情も一過性で済んでおり、何年もこのような状況に慣れてしまっているので、改善などの手間ひまがかかる活動を始める動機がわかない、という心理的な傾向の影響もあるのでしょう。

【改善活動の阻害要因】　１．現状を替えたくない心理　２．費用・時間の不足

環境的な要因

社内の改善活動にとっての最初の障害は費用および時間の不足にあります。出たとこ勝負的な開発を行っているような組織は、慢性的な赤字と時間不足に悩まされています。仕事に投入される時間は、妥当な品質を確保するために必要な時間の６０％以下なのかも知れません。現実的な問題として、このようなチームに自分たちの時間の５％を割いて改善活動を開始するように説いたところで、所詮そんな時間はどこにもないから、やはり改善活動はできないということになります。

このようなチームを復活させるためには、そのチームの予算外で動ける人材の投入が必要になるということを最初に理解しておく必要があります。チーム内に改善活動をリードする適任者がいたならば、その人材の予算を別途確保し、そのチームには背負わせないような手当てが必要となります。

経験から言えることは、改善活動によってある程度の実利が得られる見通しさえついていれば、このような初期投資の金額の数倍もの改善益はすぐにでも得られますが、ないない尽くしの中で改善活動ができないのは当り前のことであるとも言えます。

実行

◎それでも改善活動には、初期投資のヒト・カネ・時間が必要になる。

改善活動の実行コンセプト

懸命に働くではなくて賢明に働くことや、切り詰める節約ではなくリソースを有効に使うことや、継承という譲りは、個人のみならず組織やプロジェクトに永続的な繁栄をもたらします。

継承・譲りという思想を中核に据えた、ある改善活動チームの活動コンセプトは次の３点でした。

【改善活動のコンセプト】

＃１．時間の獲得

＃２．自律型開発への転換

＃３．変化即応型開発の実施

活動の概要を以下に示します。

コンセプト＃１．時間の獲得

ものごとを遂行するにおいて時間の獲得は究極の必要条件です。ヒト・モノ・カネ・情報などのリソースはなんとかなる可能性がありますが、失われた時間を取り戻すことはできません。プロジェクトの究極の成功要件は、必要時間の確保にあることを深く認識し、必要時間の獲得および無駄な時間の削減に必要なあらゆる対策を講じる必要があります。そのアクションは以下の通りです。

アクション：

① 先行調査によりリスクを排除し、後工程で発生する不始末による膨大な時間ロスを防ぐ。
要求仕様の疑問点・不明点の払拭、未経験の仕様・機能の調査・理解、プロトタイプ作成による未知の新技術領域のリスクの排除などを実行する。

② 要件定義工程において、早期の仕様凍結を目指し、顧客との集中合宿検討会を含む密接かつ頻繁な仕様検討会を行い、顧客とのコミュニケーションを緊密にすることで、要求仕様の誤解やモレをなくし、後工程で発生する膨大な時間ロスを防ぐ。

③ リスクを考慮した妥当かつ正確な見積りを行うことで、開発に必要な絶対時間を確保する。
要求仕様および開発全体に関するリスクを把握し、開発に必要な時間と予算の確保を行う。

④ 開発プロジェクト内のコミュニケーションを緊密にすることで、指示の誤解やモレをなくし、後工程で発生する膨大な時間ロスを防ぐ。毎日ベースの日次情報共有会議の実行、週次・月次のまとめ会議の実行などの直接コミュニケーションによるリアルタイムな情報の共有および関係者間の信頼の醸成を図る。

実行

⑤ 重要な機能から先に開発を行い、完成の都度、顧客と現物にて動作確認を行うことで、やり直し・修正などの膨大な時間ロスを防ぐ。 顧客に対して７日～１０日ごとの重要仕様順のリリースを繰り返し、顧客および開発チームによる共同検証を行い、不具合修正・仕様変更等は顧客との直接対話による即断・即決による対応を行う。

⑥ 上記を、より可能にするために顧客および開発チームは場所および情報の共有を行い、開発を共同して遂行する。

効果：

① プロトタイプ作成による未知の新技術領域特有の不具合のあぶり出しによる、本番開発における致命的な不具合の予防。

② 顧客との密接直接コミュニケーションによる、要件定義における相互理解・相互信頼関係の醸成。

③ 要求仕様の早期凍結をベースにした妥当かつ正確な見積りは、プロジェクトの絶対的な成功要因である必要時間の確保と必要予算の確保を確かなものにする。

④ 開発チーム内の日次・週次・月次の頻繁なコミュニケーションは、メンバー全員による問題の共有および情報の整理整頓を促進し、開発者における仕様の誤解・設計ミス・段取りミス等を大幅に削減すると同時に、開発者における開発完遂の強いモチベーションおよび信頼関係を最後まで維持させる。

⑤ 短期間で逐次リリースされた動作する現物のソフトウェアにての顧客との直接的検証・対話は、開発における後戻りや不要な修正等の無駄な作業を大幅に削減し、同時に顧客における安心感・満足感の醸成に大きく寄与する。

継承

◎プロジェクトにおける究極の制約条件は時間にある。
必要な時間の獲得・無駄な時間の削減はプロジェクトを成功に導く。

参照：第４章「４－４．獲得時間の最大化と失う時間の最小化」

【時間獲得のための改善活動のテーマ】（図４－３－２）

- ●事前準備・事前調査の推進
- ●早期仕様凍結
- ●見積り精度向上
- ●コミュニケーション活性化
- ●顧客価値優先開発の推進
- ●情報共有化の推進

ＱＣＤを成功に導くために
必要な時間の獲得

実行

コンセプト＃２．自律型開発への転換

マンネリ化したウォーターフォール型開発[31]組織を自律性のある組織へ転換させる。

アクション：

① **毎日やる**　☆課題バラシ[32]およびその対策を毎日実行する。
☆開発担当者を毎日巡回しフォローする。
☆対策の指示を毎日行う。

② **毎日聞く**　☆開発担当者の不平・不満・グチを聞く、関連部署の状況を聞く。
☆お客様の話を良く聞く。

③ **毎日見る**　☆毎日、リスク管理表・課題管理表を見る、開発担当者たちの進捗状況を見る。
☆毎日、開発担当者たちの健康状況を見る。自分の健康状況も見る。

④ **毎日まとめる**　☆変化する課題・問題を毎日、整理更新し、まとめる。
☆明日以降の必要なアクションをまとめる。

⑤ **現場に居る**　☆プロマネは心理的・地理的にいつも開発担当者のそばに居る。
☆プロマネはいつでも担当者の相談にのる。
☆プロマネは事務所よりも開発ルームで仕事をする。

⑥ **顧客の声を聞く**　☆お客様を早い時点で開発パートナーとして巻き込む。

⑦ **早くする**　☆自部署の担当ではないと判断した仕事は即刻担当部署に渡す。

⑧ **見えるようにする**
☆リーダーは、担当者・外注の仕事の範囲・進捗状況を見えるようにする。
☆必要な情報は口頭だけに頼らず、まず手書きでも良いから書面として見えるようにする。
☆スケジュール進捗表・管理表は毎日更新しておく。

これらの行動は開発者における自律性・協調性を促し、顧客との密な連携を促し、日々刻々の変化に即応するための基本的な行動のポイントとなります。

継承

◎自律性とは、
●目の前の現実や事実を自分の目で直視し、
●それが意味するところを自分自身で解釈・判断し、
●それをどのような方法で解決するのかを自分で決定し、自分で行動を起こすこと。
◎自律的な開発は、毎日の自律性に基づいた行動によって可能になる。

---

31 **ウォーターフォール型開発**　開発プロジェクトを時系列に、「要求定義」「概要設計」「詳細設計」「プログラミング」「テスト」「運用」などの作業工程に分割し、前から順次後へと開発を進める方式のこと。原則として前工程が完了しないと次工程には進めない。

32 **課題バラシ**　問題の発生原因と思われるものを可能な限り書き出し、類似したものを同一グループにまとめ、グループごとに有効と思われる対策を立案することで、複雑な問題に対する解決法を見出そうとする問題解決手法。

実行

コンセプト＃３．変化即応型開発の実施

変化即応型開発のポイントは下記の二点にあります。

１．顧客要求に対して俊敏な対応をすること。

２．開発チーム全員による意識・価値観の共有による目標の達成を目指すこと。

アクション：

① 顧客とのデイリー＆ダイレクトなコミュニケーションの実施。

② インクリメント[33]な成果物を基にした顧客と開発チームによる共同評価・検証の実施。

③ 開発チーム内のデイリー＆ダイレクトなコミュニケーションの実施。

アクションの具体例：

プロジェクトの主要な問題は人と人の間に発生するものであり、それらの問題をすばやく解決することからしかプロジェクトの成功は導き出せないという認識に基づき、顧客と開発チーム間あるいは開発チーム内において発生する、課題・問題を、成果物である現物に基づいて直接的なコミュニケーションによって毎日ベースに俊敏に解決する。

継承

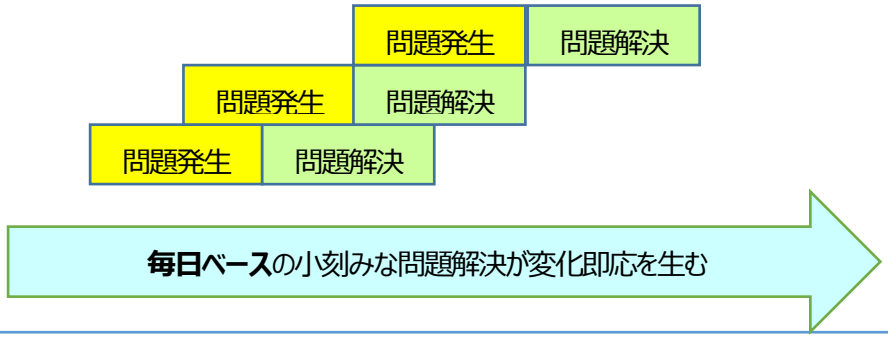
◎柔軟性に富んだ変化即応型のプロジェクトが持つ三つの行動特徴。

１．顧客とのデイリー＆ダイレクトなコミュニケーションの実施。

２．インクリメントな成果物を基にした顧客と開発チームによる共同評価・検証の実施。

３．開発チーム内のデイリー＆ダイレクトなコミュニケーションの実施。

【変化即応型開発】（図４－３－３）

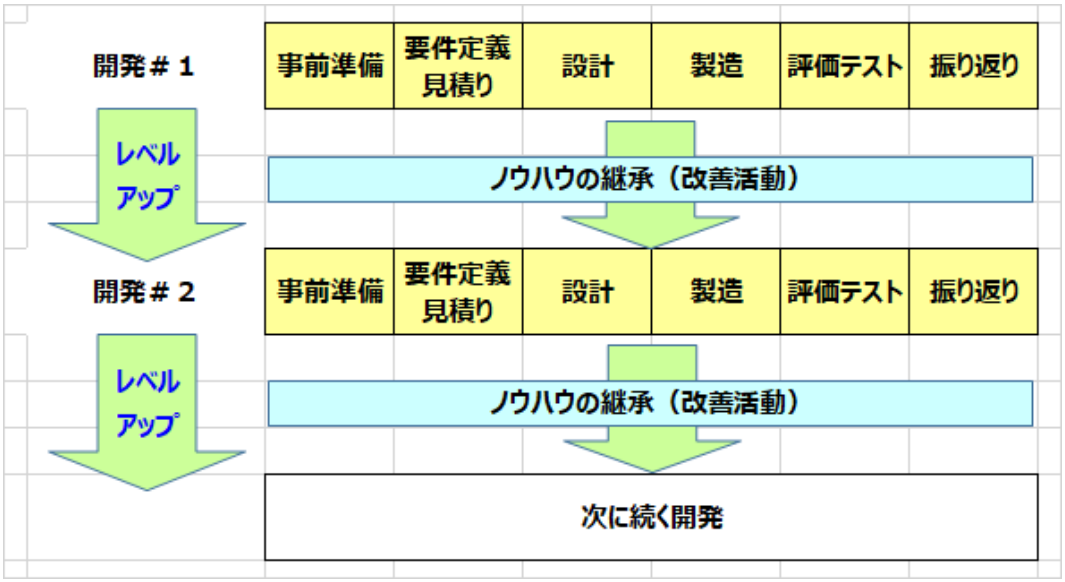


[33] **インクリメント** アジャイル開発などで逐次分割してリリースされる成果物をインクリメントな成果物と呼んでいる。アジャイル開発では、従来のウォーターフォール開発で行われているような、たくさんの要求仕様をまとめて開発・検証・リリースするのではなく顧客が要求する順に従って幾つかの仕様群に分割して順次開発・評価・リリースが行われる。

実行

譲り・ノウハウの継承・改善活動の実行は、同じことを違った言葉で表したもので、これらのことを継続して実行するプロジェクトは永続する継承循環のサイクルに入ることができ、プロジェクトのＱＣＤの成功のみならず個人および組織のスキルアップ、開発力の永続的な発展を実現することができるでしょう。

【ノウハウの継承循環】（図４－３－４）



４－４．獲得時間の最大化と失う時間の最小化

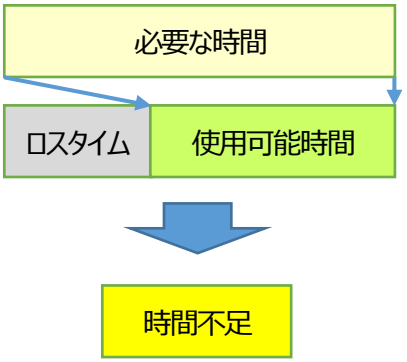
私たちが失っている時間

最初に私たちが失っている時間にはどのようなものがあるのか、プロジェクト活動の全体を通して見てみる必要があります。 私たちは下記に示すようなことが原因で多くの時間を失っています。昨日も・今日も・明日もこのようなことがあちこちで行われているのです。下記に示した事例に全く該当するものがない人など存在しないでしょう。

【ロスタイム】

① 要件定義遅れによる時間ロス
② 雑な要求仕様書による長時間の仕様調査・検討
③ 無理な納期短縮による工期不足
④ 見積りの失敗による工期不足
⑤ 他人まかせの仕事による進捗遅延
⑥ 繰り返される類似不具合
⑦ 段取りミスによる手戻り
⑧ 現物確認を怠ったためのミスによる手戻り
⑨ 口頭依存のコミュニケーションによる仕様の誤解
⑩ 情報共有不足によるミス・仕事の重複
⑪ 優先順位判断ミスによる時間ロス
⑫ 使える設計書がないことによる長時間のソースコード解析
⑬ 使える設計書がないことによる的外れな評価業務
⑭ 結論の出ない長時間・頻繁な会議
⑮ 時期外れの形式的なレビュー
⑯ 報告者の意思がない意味不明な報告書
⑰ 多数の無関係なｃｃ：メール
⑱ 仕事に関係のない長電話
⑲ 会議・会談への遅刻

【ロスタイムは取り戻せない】
（図４－４－１）



上記を見ただけでも、どれだけ多くの時間が失われているのか簡単にイメージできるでしょう。全開発時間の３～４割は軽く超えていることでしょう。

この問題はプロマネを中心としたチーム全員で取り組めば、大幅な改善が可能です。

実行

プロジェクトにおける時間の考え方

開発担当者たちの最大の悩みは時間不足です。個々の問題の表面的な原因は、要求仕様書や設計書のあいまいさ、コミュニケーションの不良や技術レベルの低さなどがあげられていますが、さらに突き詰めると「調査時間が足りませんでした」「検討時間が足りませんでした」とか「実行時間そのものが足りませんでした」というような答えが非常に多いことに気づかされます。

プロジェクトに必要なリソースとして、人・モノ・カネ・情報などが重要なものとして挙げられますが、最も重要なものは「時間」なのです。他のリソースは何らかの手段を講じればなんとか入手することは可能ですが、失われた時間を取り戻す方法などありません。実際にプロジェクトにおいて必要な時間が不足した場合に、納期を守るために行われることは、手抜きという不正行為なのです。

実行

◎プロジェクトにおける最大の悩みは、時間不足。

Ｐ．ドラッカー[34]は仕事と時間の関係について次のように語っています。

「私の観察によれば、成果をあげる者は仕事からスタートしない。時間からスタートする。計画からもスタートしない。何に時間がとられているかを明らかにすることからスタートする。（中略）成果をあげる者は時間こそが、真に普遍的な制約条件であることを知っている。」

まさにその通りで、プロジェクトはその実力に応じて開発に必要な時間が決まり、この必要な時間を獲得するための行動が見積り交渉であり、その獲得された時間を必要なタスクに割り当てる細分化作業がＷＢＳの本質なのです。もし見積りで獲得した時間がプロジェクトで必要とされる時間より短ければ、時間不足という事態に陥ることになり、プロジェクトはその出発点において失敗を運命づけられることになります。

実行

◎プロジェクトにおける普遍的な究極の制約条件は、時間である。

【時間不足が招くもの】

◎手抜きによる品質の悪化、さもなくば　**◎納期の遅延**

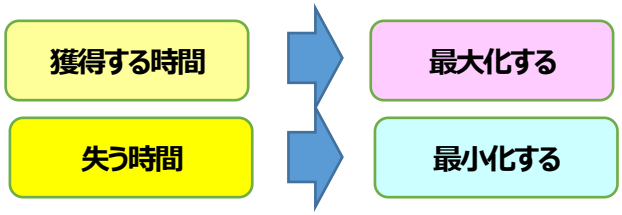
34 **Ｐ．ドラッカー**（Peter Ferdinand Drucker）　経営学者。『マネジメント』はその代表的な著書。

実行

獲得時間の最大化と失う時間の最小化

プロジェクトにおいての時間には二種類があります。一つはプロジェクトが獲得した時間、もう一つはプロジェクトで失う時間です。プロジェクトは、見積り交渉において開発期間という時間を獲得し、プロジェクトの進行中にさまざまな問題によって時間を失っています。プロジェクトを成功させるためには、そのプロジェクトに必要な時間の確保が必要であり、そのためには見積り交渉においての獲得時間の最大化と、開発行為の不手際によって失われる時間の最小化という二面性のアプローチが必要です。

【プロジェクトにおける二通りの時間】（図４－４－２）



獲得時間の最大化

獲得する時間の最大化は、主に見積り時における必要な開発期間の獲得によって達成されます。

プロマネを初めとしたマネージャは、見積り交渉において不条理な圧力に負けず、無理な短納期の要求に対しては知恵を出し条件闘争に持ち込むような、タフな交渉力および論理的な説得力が必要となります。

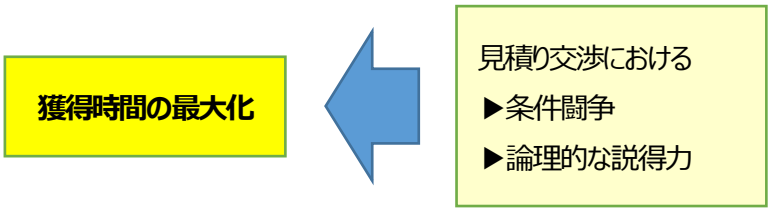
会社が戦略的な物件として不足分を補う約束をしていない限りは、コスト競争に勝つためという理由で、本当の生産性を上げる努力もしないまま、低価格・短納期の見積りを出すべきではありません。

他社との競争に勝てるような見積りが提示できるようになるには、失敗や無駄の排除および効率的な開発の実践による、生産性の向上などの継続的な改善活動を行う必要があります。改善活動の実践は見積り交渉時における論理的な説得力およびタフな交渉力の源泉ともなります。

実行

◎見積り交渉にて妥当な開発期間・費用を獲得するためには、
普段の改善活動の実践による、論理的な顧客説得力の強化が必要。

【獲得時間の最大化】（図４－４－３）



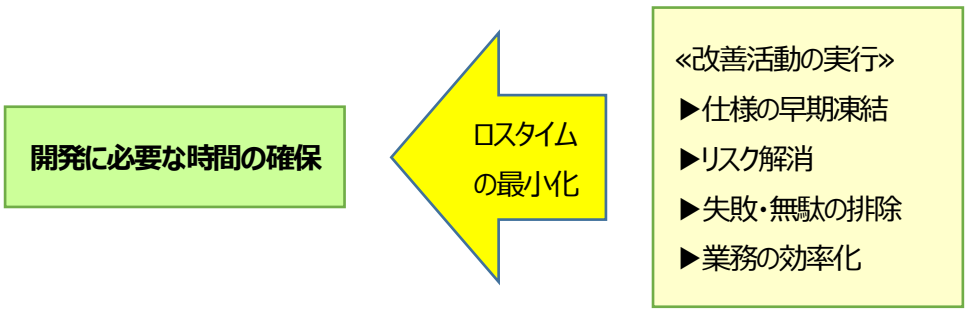
失う時間の最小化

失う時間の最小化に貢献する具体的な行動としては以下のものがあります。

① 要求仕様の早期凍結を実現すること。
② 開発初期の時点で大きなリスクを解消しておくこと。
③ 同じ失敗を繰り返さないこと。
④ 無駄を排除すること。
⑤ 仕事自体の効率化を行うこと。

いつまでも決まらない二転三転する要求仕様が、失われる時間の代表的なものです。顧客から要求仕様が出て来るのを待つのではなく、提案型のアプローチのもとに、集中的な要求仕様検討会を実行することで、早期の仕様凍結は可能になります。少なくとも基本設計着手前には、要求仕様の骨子の決定が必要であり、詳細設計着手前には、詳細レベルの要求仕様の決定が必要です。

【ロスタイムの最小化】（図４－４－４）



プロジェクトにおける主な時間効率化方法

プロジェクトにおいては、さまざまな効率化施策が必要となりますが、いずれにしても時間効率性を高めることに効果のある施策は、同時に品質やコストの改善にも有効に働きます。下記は実際のプロジェクトにおいて必要とされる重要な効率化方法です。

目標の明確化および集約化

1．顧客との密接なコミュニケーションを通して、要求仕様の明確化および開発項目の優先度付けを行なうこと。顧客価値のないあるいは低い開発を防止し、顧客価値のある開発に集中すること。

2．チーム内の短時間日次会議の繰り返しを通して、目標の明確化および全員による共有化、開発情報・問題情報の共有化、行動の方向性の統一化を図ること。

【目標の明確化および集約化】

- 要求仕様の明確化
- 開発項目の優先度付け
- ＱＣＤ目標の明確化・共有化
- 問題情報の共有化
- チーム行動の方向性の統一化

変化即応的な組織運営

1．科学的データ・思考に基づくプロジェクト運営を行なうこと。情緒的な人間関係に片寄らず、非人間的な運営に傾かず、合理的かつ妥当性のある組織運営を行なうこと。

2．多重請負構造を避け、役割りの丸投げを禁止し、参加組織数の削減に心がけることで、役割分担の明確化を図り、情報伝達性・情報正確性・組織統合性を向上させること。

3．プロジェクトを構成している複数の組織間の情報連携、活動連携を密接にすること。

【変化即応的組織】

- 合理的・妥当性のある組織行動
- 多重請負構造の回避
- 役割丸投げの禁止
- 役割分担の明確化

実行

リスク管理・プロセス管理の実行

1．リスク管理により、予測される全ての失敗を予防すること。

2．あるべきプロセスの確実な実行により、手抜き・実行忘れなどの不要な失敗を防止すること。

【リスク管理・プロセス管理】

- ●リスクの解消
- ●手抜き・実行忘れの防止

ドキュメントベース開発の実行

1．基幹ドキュメントの質の向上およびメンテナンスを励行することで、ノウハウの蓄積および継承の最大化を図ると同時に、ドキュメントベースによる強力な情報共有化を通した、正確かつ無駄のない開発を実行すること。

2．使用技術用語の統一化による、顧客・開発間の意思疎通レベルの向上を図ること。

【ドキュメントベース開発】

- ● ドキュメントの質の向上
- ● ドキュメントによるノウハウの蓄積と継承の拡大
- ● 情報共有化による無駄のない開発
- ● 技術用語統一化によるコミュニケーションの質の向上

柔軟なシステム構造の考慮

スピードと柔軟性を実現する高メンテナンス性[35]を持ったシステム構造体、すなわち高メンテ領域と低メンテ領域の分離構造の実現等によって、仕様変更による他の領域への影響度を最小限に抑える。

【柔軟なシステム構造】

- ● 高メンテナンスソフトウェア構造による、仕様変更・追加に対する開発スピード性と柔軟性の向上

---

35 **メンテナンス性** 仕様変更の容易性が高いこと。いわゆるスパゲッティプログラムは、低メンテナンス性の代表例。

実行

ノウハウ循環サイクルの実現

1．振り返り会議を通して失敗の教訓に学び、同様な失敗を組織的に防ぐと同時に、ノウハウの蓄積および継承を行なうこと。

2．改善活動の常態化による、継続的な効率化の進展および部下の育成を図ること。

3．知識インフラとしての顧客情報・技術情報の収集、学習、教育を継続的に実行すること。

4．指示待ちではなく、自分の頭で考え・判断・行動する自律的人材を育成すること。

【ノウハウの循環サイクル】

- 失敗の教訓に学び、類似の失敗を予防
- ノウハウの蓄積と継承による、好循環開発サイクルの実現
- 改善活動による、効率化および自律的な人材の育成

実行

◎知恵と工夫と努力により、時間は手に入れることができる。

参照：第４章　４－３．改善活動の実行◎改善活動の実行コンセプト▷コンセプト＃１．時間の獲得

【実行工程のまとめ】

実行工程の最大の目的は、ＱＣＤ目標の達成でした。本文の順に沿って振り返ってみます。

実行

【データドリブン開発の実行】
◎基本的なデータはＱＣＤの三つ
◎ＱＣＤそれぞれの過去・現在・目標の三つのデータで開発を進める
◎データドリブン開発を支える三つの開発コンセプトは、
・ドキュメントベース開発
・プロフィットドリブン開発
・優先順位ベース開発

開発の基本はＱＣＤの過去・現在・将来の目標値などのデータに基づいた実行が必要です。
正確なドキュメントによる品質目標値の達成、仕事の優先順位の正しい認識による納期・生産性目標値の達成、およびプロフィットドリブン志向によるコスト・利益目標の達成により、プロジェクトを成功に導くことが可能になります。

実行

【優先順位ベース開発】
◎無駄な開発の排除のために
① 顧客価値のない要件を排除すること。
② 顧客価値の低い要件の優先順位を下げること。
◎有効な開発に要するエネルギーの最小化を図るために
① 顧客の要求と開発行為を一致させること。
② 顧客価値の優先度の順・必要とされる時期に成果物を提供していくこと。
◎ＱＣＤに優先順位を設けてはいけない

最大の時間効率性を発揮するために、優先順位の判断と行動は必ず必要です。判断に迷った場合は知恵のある人に相談すべきでしょう。但しＱＣＤに優先順位は付けられないということを強く認識しておく必要があります。

実行

【改善活動の実行】

◎改善活動は、開発業務の中核業務であり、暇な時についでにやるようなものではない。

◎良い仕事＝改善活動＋ルーチンワーク

◎改善活動の基本コンセプト：共に成長し、持てる智・財を継承し、永続的な繁栄の循環を実現すること。

ＱＣＤ目標の達成および開発者のスキルアップを確実に実現するためには、開発行為そのものが改善行動である必要があります。いつもと同じようなルーチンワーク的な作業を続けることは、開発という名前にはふさわしくないでしょう。

実行

【獲得時間の最大化と失う時間の最小化】

◎プロジェクトにおける最大の悩みは、時間不足。

◎プロジェクトにおける普遍的な究極の制約条件は、時間である。

◎見積り交渉にて妥当な開発期間・費用を獲得するためには、普段の改善活動の実践による、論理的な顧客説得力の強化が必要。

◎知恵と工夫と努力により、時間は手に入れることができる。

プロジェクトの最大の問題は時間不足にあります。時間は究極の制約事項であり、開発者の行動もＱＣＤの到達レベルも限定されてしまいます。改善活動として、このテーマに取り組むことはプロジェクトおよび開発メンバーに多くの恩恵をもたらすでしょう。

以上で取り上げた実行のプロセスにおけるテーマについては、開発の規模に関係なく、すべてプロマネ主導で実行される必要があります。

第５章　管理工程　タイムマネジメント

５－１．要求仕様の変更管理

一旦、顧客と開発側において開発すべき仕様および開発条件（開発費・開発期間・品質条件）などが合意されたとしても、その後に顧客側ないしは開発側の都合によって変更や追加の要求が出された場合、最初に約束した内容を守ることが難しくなってしまいます。特に仕様の度重なる変更や追加は開発費・開発期間・品質に重大な悪影響を与えてしまいます。

ベースラインの意味

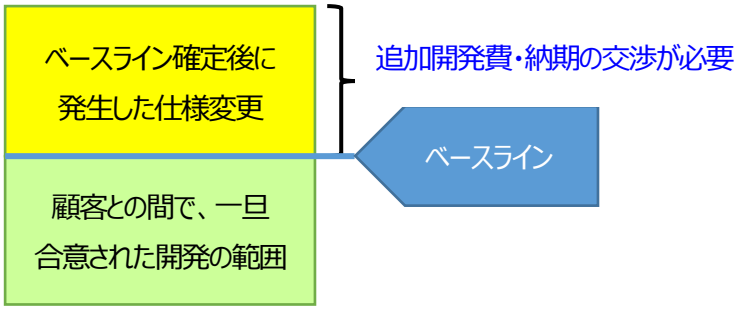
見積り回答によって顧客と約束した、開発費・開発スケジュール・品質条件および開発仕様の内容・範囲のことをベースライン[36]と呼び、このベースライン確定後の開発条件の変更に対する顧客との交渉を伴う処置を変更管理と呼んでいます。

開発側においては、大きな変更によるＱＣＤ等に対する重大な悪影響を避けるために、最初に約束したもの（ベースライン）とその後に発生したもの（変更・追加）を分けて処理する必要があります。追加・変更分は基本的に追加開発費・追加開発期間で処理される必要があります。

管理

◎ベースライン（基線）とは、
プロジェクトマネジメントにおいては、顧客と開発側で合意した成果物の範囲および基準の事を指す。両者間で約束した仕様とそのＱＣＤに関する基準は重要なベースラインとなる。

【ベースラインと仕様の変更管理】（図５－１－１）



36 **ベースライン**　見積り回答によって顧客と約束した開発費・開発スケジュール・品質条件および開発仕様の内容・範囲のこと。

管理

変更管理が行われにくい理由

変更管理が行われにくい理由として最初に考えられることは、開発のスコープ・予算・期間を決定してしまうベースラインの確定そのものを顧客側が嫌うということがあります。

一般的に顧客側は限られた予算と期間の中で、できるだけ多くの仕様を盛り込みたいと思っていますが、ベースラインの確定はそれに対しての縛りとなってしまいます。

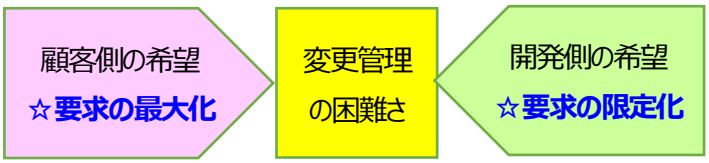
顧客側の考え方は、できるだけ良いものをできるだけ安く手に入れたいという所から出ているのでしょうが、そのように都合の良いことがいつでもあるわけがなく、相応の価値あるものに相応の対価を払うという健全なビジネスの基本からは逸脱する考え方と言わざるを得ません。

余談になりますが、優れたビジネスを行っているある顧客は「品質にはそれ相応の対価を支払う」と言っており、その顧客の値引き要求に対する回答に対しても削減の根拠の提示を求められました。根拠のない値引き提案など受け付けられないと言うことなのです。

管理

◎健全なプロジェクトには、ベースラインの確定が必要である。

【変更管理の困難さ】（図５－１－２）



要求仕様の変更管理不在がもたらす災い

要求仕様のベースラインの確定も変更管理も実行されないならば、顧客側は白紙委任状[37]を手にしたのも同然で、早期の仕様まとめの動機も失われ、顧客のあちこちの部署からさみだれ的に集まってくる要求をさみだれ的に開発会社に投げ続けることになります。これが二転三転する要求仕様の本当の原因なのです。要求仕様を考える能力がないのではなく、要求仕様を決められた時期までにまとめる意思がないだけのことなのかも知れません。

このように際限なく続く仕様の変更や追加は、ついにはプロジェクトの処理能力を超えてしまい、多くの手戻り作業や追加開発により、膨大な時間を失い、品質劣化、コスト高、生産性低下を招いてしまいます。

この結果は顧客側にもブーメランのように戻ってくることになり、顧客は納期遅延や障害多発に見舞われ、最悪の場合は顧客のビジネス自体が止まってしまうこともあります。

ベースラインの確定も変更管理の実行もなされないプロジェクトは、顧客側・開発側双方において致命的な結果を招くという認識が必要です。

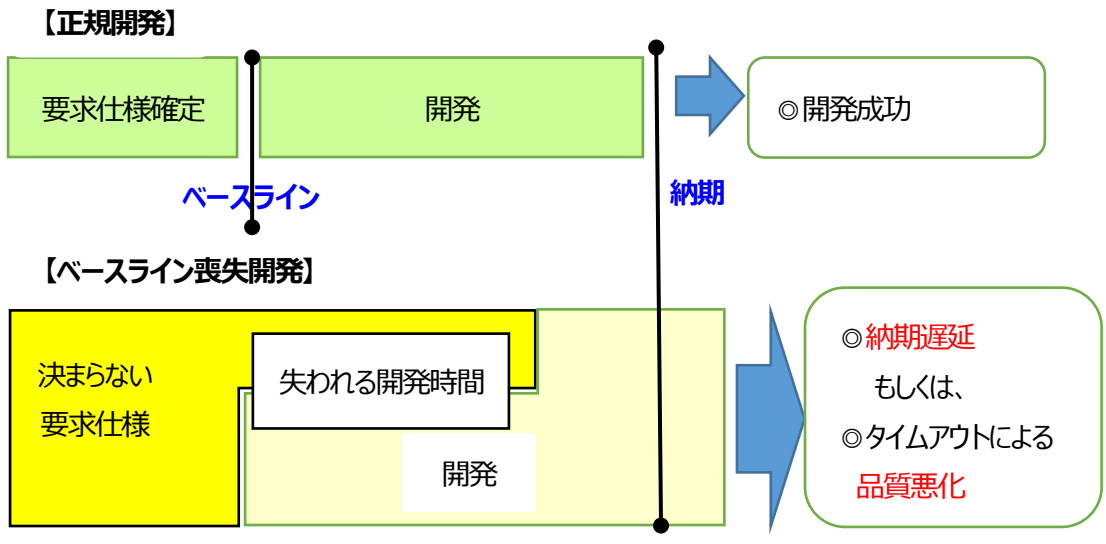
管理

◎ベースラインの確定・変更管理の不在がもたらすリスク

☆プロジェクトの失敗：ＱＣＤの大幅な未達

☆顧客ビジネスの停止：市場障害の多発

【正規開発とベースライン喪失開発】（図５－１－３）



37 **白紙委任状** ものごとの決定権を相手に委ねてしまうことの例え。

管理

仕様変更管理表がもたらす効果

安易な仕様変更にしっかりとした歯止めをかけるためには、仕様変更管理表の運用が不可欠です。

この管理表の主な目的は、仕様変更の影響を管理し、顧客側の無定見な仕様変更・追加を抑制することにありますが、特に顧客と開発側で同意した仕様凍結日以降に発生する仕様変更を厳重に管理することが重要であり、同意された仕様凍結日以降に発生した変更・追加仕様は基本的には別途見積り（追加コスト・追加開発期間）であることを顧客側が明確に認識するために使用されます。 仕様変更管理表の運用は下記のような効果をもたらします。

【仕様変更管理表の効果】

1．放縦・無定見な仕様変更の防止
2．顧客・開発側両者に対する仕様凍結期日の厳守化（開発仕様のベースラインの確定）
3．情緒的な要件定義工程のプロセス化
4．仕様変更の質、量および費用の厳格な管理

管理

◎ベースライン確定後の仕様変更・追加分は別途見積りとする交渉が必要。

仕様変更管理表には、管理番号、発生日、対応期限、対応日、対応状況、業務名、変更内容、変更区分、修正担当、影響度範囲、難易度、リリースバージョン、見込み対応工数、実績対応工数などを記録し、仕様の変更状況を明確にしておく必要があります。仕様変更管理表のフォーマットは次のようなものになります。

【仕様変更管理表】（表５－１－１）

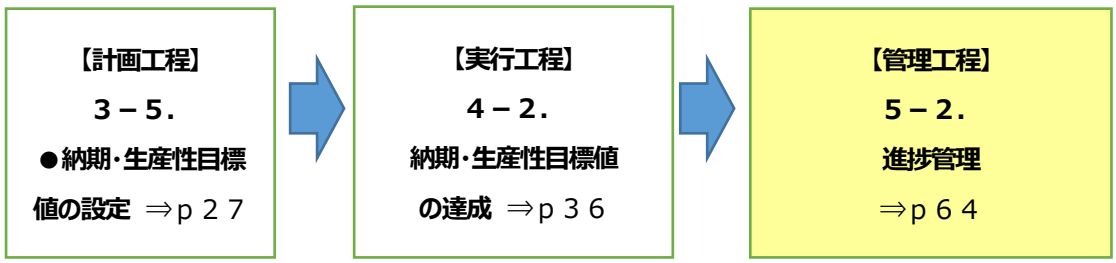
【仕様変更管理表】

No.	管理番号	発生日	対応期限（ベースライン）	対応日	対応状況	業務名	変更内容	変更区分	修正担当	影響度範囲	難易度	リリースバージョン	対応工数（見込）	対応工数（実績）
1	PJ-001										A			
2											B			
3											C			
4														
5														
6														
7														
8														
9														
10														
合計														

拡大図表は、付録図表２．を参照のこと。

５－２．進捗管理

これまでの納期・生産性管理に関する解説の流れを図示すると下記のようになります。



第３章の３－７．においては納期・生産性目標値の設定方法について、第４章の４－５．においては納期・生産性目標値の達成活動について解説をしました。本章においてはこれらの設定された納期・生産性目標値の達成状況をモノ・カネ・ヒトの三つの視点による進捗管理で見て行くことにします。

第３章　計画工程「３－４．スケジュールの作成」においてＷＢＳに基づいた開発スケジュールを作成しましたが、計画されたスケジュールを予定通りに進捗させるための方法について解説を行います。

プロジェクトの進捗管理といえば、モノという成果物を生み出す要件定義・設計・製造の進捗ばかりに目を向けがちになりますが、プロジェクトを成功裏に完了するためにはヒト・モノ・カネの三点すべてに対する進捗管理が必要になります。

開発の半ばにおいてプロダクト成果物の進捗がオンスケジュールであったとしても、予算の消費が８０％にもなっていればプロジェクトは大赤字になってしまうことは確実で、また開発者の平均残業時間が月１００時間を超えてしまっていればプロジェクトの遂行は困難になってしまいます。

納期・利益第一主義的風潮が強い状況下にあって、モノの進捗管理一辺倒では顧客が納得できるＱＣＤの達成は難しいでしょう。ヒト・モノ・カネのリソースが健全に使用されて初めて妥当なＱＣＤ目標が成立し、顧客の満足が得られる製品の開発が可能になります。

管理

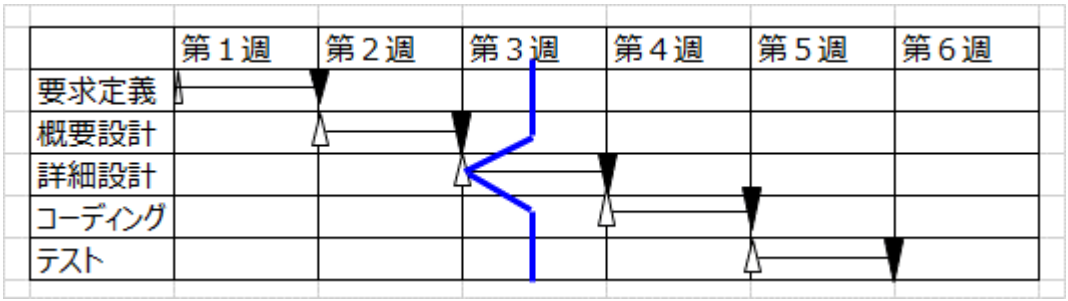
◎進捗管理の三点セット

１．モノの進捗管理　　：プロダクト成果物進捗管理
２．カネの進捗管理　　：予算消費進捗管理
３．ヒトの消耗度管理　　：残業時間管理

モノの進捗管理（プロダクト成果物進捗管理）

ソフトウェア開発の進捗管理には、一般的に下記に示したようなガントチャート[38]が用いられています。ガントチャートは視覚的に分かりやすく、大まかな進捗を管理するのには便利な管理表だと言えます。

【ガントチャート】（表５－２－１）



ガントチャートは日程表に進捗線を入れたもので管理されますが、この表ではタスクの正確な進捗を見ることができません。進捗線が１週間遅れの様に記入されていたとしても、本当に１週間なのかどうか記入した本人ですら分からないのが実状なのです。ひょっとしたら２週間遅れなのかも知れません。

「ソフトウエアの開発スケジュールは、納期の直前までは順調に進む（木下恂著、ソフトウエアの法則）」というような警句にもある通り、報告を受けたガントチャート上の最後の２週間遅れが、一か月を経過しても解消されないこともあります。

ガントチャートは大まかな進捗を見るのには適した表現方法ですが、おおまかさを補うために次に示した機能モジュール進捗管理表[39]の併用が有効です。機能モジュール進捗管理表は縦軸に開発タスクを最小レベルまでブレークダウンしたソフトウエアモジュール名を記載し、横軸にはそのプログラムの見込みサイズ、開発各工程の終了予定日・実績日を記入して予定日との差異を管理するものです。最小プログラムモジュール単位までブレークダウンされているため、主観の入る余地が少なく正確な進捗が見えやすいものとなっています。

機能モジュール進捗管理表は、いわばＷＢＳ的なモノの進捗管理ないしは開発物の単品管理[40]ともいえるものです。

機能モジュール進捗管理表のサンプルを次に示します。

---

38 **ガントチャート** ガントチャートは日程表に進捗線を入れたもので、視覚的に分かりやすく大まかな進捗を管理するのには便利な管理表だと言えます。

39 **機能モジュール進捗管理表** 縦軸に開発タスクを最小レベルまでブレークダウンしたソフトウエアモジュール名を記載し、横軸にはそのプログラムの見込みサイズ、開発各工程の終了予定日・実績日を記入して予定日との差異を管理するもので、いわばＷＢＳ的なモノの進捗管理ないしは開発物の単品管理ともいえる。

40 **単品管理** モノの無駄をなくすために、商品を一品ごと、販売・仕入・在庫の管理を行う手法。コンビニの単品管理は好事例。

管理

【機能モジュール進捗管理表（部分）】（表５－２－２）

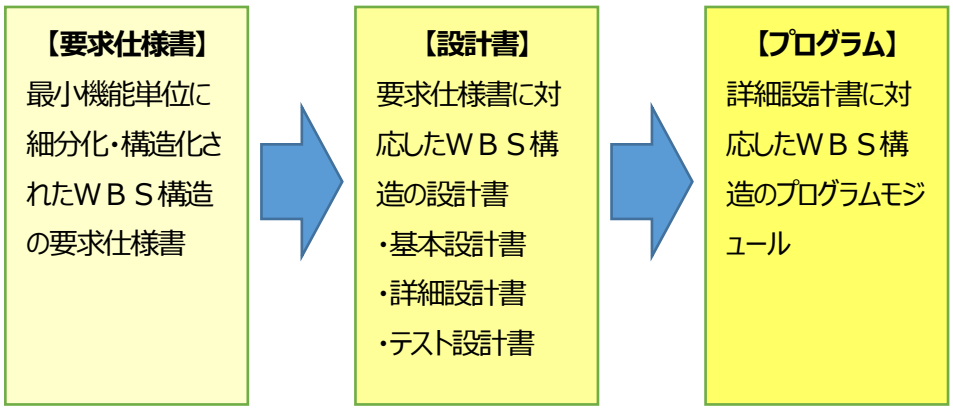
業務	プログラム名称	ＥＸＥ，ＤＬＬ、ＯＣＸ等名称	開発Ｖｏｌ（Kbyte）		担当者	開発状況	コーディング（Ｃ）		単体評価（Ｕ）	
			見込	実績			完了（予定日）	完了（実績日）	完了（予定日）	完了（実績日）
	ＸＸＸ受信プロセス	ｘｘｘｘｘ.exe	50	60	A、B	C	6/30	6/30	8/15	8/15
	ＸＸＸ送信プロセス	ｘｘｘｘｘ.exe	45	40	A、B	C	6/30	6/30	8/15	8/15
	通信プロセス	○○.exe	25	30	A	U	6/30	6/30	7/15	7/15
	リカバリー応答プロセス	○○RECOV.exe	35	30	A、B	U	6/30	6/30	7/15	7/15
	△△受送信プロセス	△△RECV.exe	25	20	A	K	5/30	5/30	6/15	6/15
	△△送受信プロセス	△△SEND.exe	20	30	A	K	5/30	5/30	6/15	6/15
	△△起動問合せ	△△INQ.exe	30	10	B	K	5/30	5/30	6/15	6/15
	ＦＴＰプロセス	○○FTP.exe	20	30	C	K	5/30	5/30	6/15	6/15
	最小モジュール単位までブレークダウンする		量的把握の見込み／実績を行う		担当者の明確化		工程別進捗を予定／実績日程にて行い。色彩にても直感しやすくする			

詳細拡大図は付録図表の「３．機能モジュール進捗管理表」を参照のこと。

管理

◎モノの進捗管理は、単品管理の思想で行うこと。

【最小機能単位で作成管理される開発成果物】（図５－２－１）



カネの進捗管理（予算消費進捗管理）

カネすなわち開発費の消費状況の管理は原価管理表だけでは不十分です。原価管理表を見て、今の時点において設計業務にいくら開発費を使っていくら残っているのかを即答できるプロマネはいないでしょう。

開発費の見積りは、業務の種類別すなわち開発工程別の作業に分解されたものの見積りの合計額であり、例えば要件定義工程・設計工程・製造工程・評価テスト工程それぞれに必要な費用が見積もられています。プロジェクトが健全に運営されるためには、各工程の業務がその見積り予算を目標に遂行されているか否かをモニターする必要があります。

工程別の費用配分・期間配分の例を次に示します。

【開発工程における工数比率・期間比率】（表５－２－３）

工程	要件定義	基本設計	詳細設計	製造	総合テスト	合計
工数比率	15.0%	13.0%	12.0%	40.0%	20.0%	100.0%
期間比率	25.0%	15.0%	12.0%	25.0%	23.0%	100.0%

出典：日経ＩＴプロフェッショナル２００２年１０月（ＩＢＭ：１４０頁）

調査対象：小規模プロジェクト（１００人以下のプロジェクト、１０数万ステップ規模）

注．筆者にて改変または推定した部分：

①原典において表記の外部設計、内部設計はそれぞれ基本設計、詳細設計の表記に置き換えました。

②原典における統合テストおよびシステムテストは、総合テストとしてまとめました。

③単体テスト・結合テストは製造に含まれているものと推定しました。

上記のデータはほぼ現実の経験値と一致しています。但し未経験分野の新規プロジェクトにおける総合テスト工数比率は３０％程度に膨らむ可能性があります。

読者において現在担当されているプロジェクトにおける数値と対比してみてはいかがでしょうか。何か別の気づきがあるかも知れません。

例えばあるプロジェクトの見積りが上記のデータのようであった場合で、要件定義工程の実績工数比率が２５％になろうとしていた場合は、要件定義工程が遅延していることになります。その他の工程についても同様であり、見積り時の数値から大きく乖離している状況が発生しそうな場合は、その工程に何らかの異常事態が起きている証拠であり早急な是正措置が必要になります。

開発業務の健康状態は開発費の消費状況にはっきりと表れてきます。

管理

各工程における開発予算と消費実績を管理するためには、通常の原価管理表とは別に次のような工程別原価管理表である予算進捗管理表[41]が必要になってきます。

【工程別予算進捗管理表】（表５－２－４）

	工程	要件定義	基本設計	詳細設計	製造	総合テスト	合計
予算	金額						
	%						
実績	金額						
	%						
差額	金額						
	%						

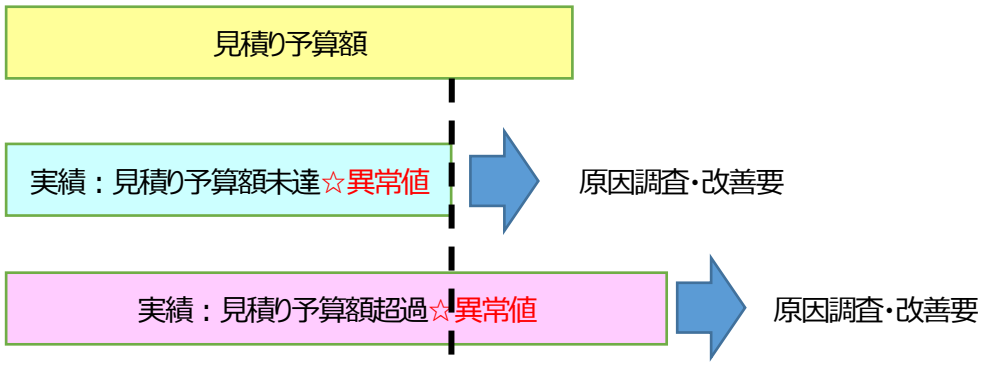
各工程の予算と実績に乖離が出そうな場合には、その工程に問題があるということで、予算オーバーのみならず予算未達の場合においても、その原因を調べ早急な対応が必要となります。

少しくらいの乖離ならばいいかと放置していると、詳細設計が終わるころまでに、本来は全予算の４０％程度の消費であるべきところが８０％も消費していたというような状況に陥る危険性もあります。

管理

◎開発費の消費状況からプロジェクトの健全性を見るためには、
工程別の予算進捗管理が必要になる。

【予算消費の健全性確認】（図５－２－２）



41 **予算進捗管理表** 各工程の予算と実績の乖離状況を把握することで、資金面におけるプロジェクトの健全性をチェックする管理表。

ヒトの消耗度管理（時間外労働時間管理）

開発を進める主体はあくまでも開発者であるヒトにあります。開発費の健全性を管理するために原価管理表や工程別予算進捗管理表を用いましたが、ヒトの健全性を確保するための管理も必要です。

進捗管理の横軸は時間や期間ですが、開発時間の経過と共に開発者たちの疲労度は間違いなく増加していき、過重労働を放置したままではプロジェクトは必ず行き詰ります。
ヒトの消耗度管理は進捗管理の中では、ある意味で最も重要な管理表だとも言えます。中核の開発者が倒れてしまえば、全ての管理業務は全く機能できなくなります。

ヒトの消耗度管理は、基本的に毎月の時間外労働時間の管理をするということですが、単に６０時間や８０時間を超えないように管理するだけではなく、個々人の体調を良く把握した上で制御する必要があります。プロマネにおいては自分自身の健康管理も含め、メンバー全員に対して以下のような消耗度の管理が必要です。

① 全員の健康状態について常に気を配ること。
② 危ない状況の人に対しては即刻休ませること。
③ 欠員が発生する前に他からの人材の補充を行うこと。
④ 法定・協定における時間外労働はしない、させないこと。
⑤ 超過残業や休日出勤に対しては、直後に必ず代休や半日休暇を設けること。
⑥ 特定の人間に負荷が集中しないように柔軟に人員体制を組み替えること。

上記のような施策を可能にするためには、本書の全てにおいて言及しているような、妥当性に基づいた合理的なプロジェクト運営を行う必要があります。プロジェクトから一人たりとも犠牲者を出さないという心構えが必要です。これはプロマネの人間としての最大の義務だと言えます。

管理

プロジェクトメンバーの労働負荷状況の管理表[42]の例を次に示します。

【労働負荷状況管理表】（表５－２－５）

プロジェクトメンバー氏名	先々月残業時間	先月残業時間	当月残業時間	３か月合計	１ヶ月平均残業時間	労働負荷レベル	労働負荷状況の理由	負荷軽減対策

労働負荷レベル

５：非常に過度　：残業８１時間／月以上
４：過度　：残業８０時間～４１時間／月
３：普通　：残業４０時間～２１時間／月
２：軽度　：残業２０時間～１１時間／月
１：非常に軽度　：残業１０時間～０時間／月

拡大図は「付録図表４．労働負荷状況管理表」を参照のこと。

【負荷ブラック度判定表】（表５－２－６）

プロジェクトメンバー氏名	先々月残業時間	先月残業時間	当月残業時間	３か月平均
A	4	5	5	5
B	3	4	4	4
C	2	3	4	3
D	2	3	4	3
人員数	4	4	4	4
ブラック人数	1	2	4	2
ブラック度%	25%	50%	100%	50%
是正の要・不要	不要	要	要	要

[42] **労働負荷状況管理表**　開発者の人的な消耗度を毎月の時間外労働時間によって管理するもの。

【管理工程のまとめ】

管理工程の役割は、開発状況のモニタリングおよびその正常化のためのコントロールにあります。本章では、開発の基本的なベースラインである顧客との約束に基づいた要求仕様の適切な変更管理およびその要求仕様に基づいた開発成功の条件の一つであるスケージュールに対するモニタリングおよび正常化コントロールについて述べてきました。

これらの活動を整理すると以下のようになります。

要求仕様の変更管理

管理

◎ベースライン（基線）とは、
プロジェクトマネジメントにおいては、顧客と開発側で合意した成果物の範囲および基準の事。両者間で約束した仕様とそのＱＣＤに関する基準は重要なベースラインとなる。

管理

◎ベースラインの確定・変更管理の不在がもたらすリスク
☆プロジェクトの失敗：ＱＣＤの大幅な未達
☆顧客ビジネスの停止：市場障害の多発

管理

◎ベースライン確定後の仕様変更・追加分は別途見積りとする交渉が必要。

管理

進捗管理

管理

◎進捗管理の三点セット

1．プロダクト成果物進捗管理（モノの進捗管理）

2．予算消費進捗管理（カネの進捗管理）

3．残業時間管理（ヒトの消耗度管理）

管理

◎モノの進捗管理は、単品管理の思想で行うこと。

管理

◎開発費の消費状況からプロジェクトの健全性を見るためには、工程別予算進捗管理が必要になる。

第６章　振り返り工程　タイムマネジメント

６－１．プロジェクト完了報告書

プロジェクト完了時に実行される振り返り会議（ラップアップ・ミーティング）においては、仕様変更管理、品質管理、コスト管理、進捗管理それぞれの目標と実績の差の理由および今後の課題を総括し、次に続く開発への申し送り情報としてプロジェクト完了報告書にまとめます。

プロジェクト完了報告書に記載される主な内容を下記に示します。

仕様変更管理の考察　（表６－１―１）

仕様変更管理の考察

No.	管理番号	発生日	対応期限（ベースライン）	対応日	対応状況	業務名	変更内容	変更区分	修正担当	影響度範囲	難易度	リリースバージョン	対応工数（見込）	対応工数（実績）
1	PJ-001	xx.09.31	xx.10.31								A			
2											B			
合計														
考察	1．対応工数の見込みと実績の差異の理由および今後の改善課題 2．仕様変更管理全般についての反省													

顧客と約束した要求仕様のベースラインが確定した後の仕様変更は極力避けたいものですが、現実問題としては変更要求が発生してくることが少なからずあります。プロジェクトを終了するにあたって、仕様のベースライン確定後にどれくらいの仕様変更量があったのかを定量的に把握し、あまりにも大きい場合は次の開発においての顧客との仕様検討のやり方を改善する必要があります。

拡大表は「付録図表５．プロジェクト完了報告書●仕様変更管理の考察」を参照のこと。

振り返り

進捗管理の考察（表６－１－２）

進捗管理の考察	各工程における進捗の目標と実績の差異の理由およびに今後の改善課題
要求定義	
概要設計	
詳細設計	
コーディング	
テスト	
進捗全般に関する反省	

スケジュールはたとえ期日が形の上で守られたとしても、品質目標が達成されていなければ何の意味もなく、スケジュールが守られたとは言えません。

進捗管理の考察においては各工程における進捗遅延の本当の原因についての考察が必要であり、その改善対策を前記の３項目の考察から導かれた改善対策と合わせて次の開発に生かす必要があります。

振返

◎プロジェクト完了報告書は、そのプロジェクトのあるがままの結果を示すと同時に、
次に続くプロジェクトに活きる目標を提供する必要がある。

振り返り

６－２．やる気を起こす～モチベーションの喚起

社員のモチベーションを上げるために多くの会社は何らかの褒章制度を持っていますが、それは社員たちのモチベーションの向上には余り役に立ってはいないようです。褒章をもらいたいために頑張る人などほとんどいないと思われます。また人は自分がすでに持っているようなものを褒美としてもらったとしても、大して喜びもしないでしょう。現代の世の中は物であふれており、余程貴重で高価なもの以外はみんなが既に持っているものばかりです。多くの人は物に飢えている訳ではなく、心を充足させてくれる事に飢えているのだと思われます。

人々が最も心的に価値をおいているものは、将来に対する安心安全や希望でしょう。組織の大小を問わずリーダーの重要な役割の一つは、人々が希望を持てるような施策を実行することです。この先良いことが起こりそうだという実感は間違いなく人にやる気を起こさせます。
人々のやる気によって良い仕事が行われ、良い製品が生み出され、同時に利益も生み出されます。

現代のＩＴプロジェクトにおいて最も不足しているものは「時間」であることに間違いはないでしょう。多くの開発者を最も苦しめているのは時間不足です。妥当な開発を実行するために必要な時間が全く不足しているのです。

開発者たちの時間を確保するために必要なことは次の二つだけです。
１．獲得する時間の最大化
２．失う時間の最小化
詳細は第４章「４－４．獲得時間の最大化と失う時間の最小化」をご参照ください。

プロマネにおいては、この二つを満足させる施策を実行することで、深夜残業や休日出勤を撲滅し、開発者たちの心からのモチベーションを喚起することができるでしょう。

振返

◎やる気やモチベーションの喚起は、
開発者の時間不足の解消によってもたらされる。

【振り返り工程のまとめ】

振り返りのプロセスは、より良い明日を迎えるためには必須のものです。本章では、プロジェクト完了報告書における、仕様変更管理結果の考察ならびに進捗管理結果の考察について順に述べてきました。

これらの内容を整理すると以下のようになります。

振返

【プロジェクト完了報告】

◎プロジェクト完了報告書は、そのプロジェクトのあるがままの結果を示すと同時に、次に続くプロジェクトに活きる目標を提供する必要がある。

振返

【モチベーションの喚起】

◎やる気やモチベーションの喚起は、開発者の時間不足の解消によってもたらされる。

付録図表 目次

1．DMAICシート簡易版：第４章　４－３．●改善活動計画書、図４－３－１

【ＤＭＡＩＣシート簡易版】

承認印			

DMAICシート(簡易版)

テーマ：仕損費コストの削減

報告年月日　：
ＰＪリーダー名　：
ＰＪメンバー名　：

Define　＊問題の明確化

有るべき姿・こうありたいと思う状態

１．ソフトウェア製品における市場品質の向上

２．上流工程での品質確保

現在の状態

ＸＸ年下期仕損費　：xxxx千円

問題点（ＰＪが取り組む課題・テーマ）

１．デザインレビューだけでは上流工程での品質の確保は難しい。（プロセス及びリスクに関するレビューが不足。）

２．単純な不具合（プログラム製造段階での不具合）が相変わらず多い。

Analyze　＊問題を派生している根因の追及

１．注文書発行の遅延等で、本来のタイミングでの公式レビューが実施されておらず、問題点が発見されても後戻り出来ない工程になっている事が多い。又、短時間の公式レビューでは、開発プロセス及び リスクに関する細かなレビュ ーが出来ない為、公式レビューを補完する他のレビューが必要。

２．委託外注からの成果物受入検査を、総合テストで行っている為、総合テスト時に単体テスト不足による不具合が多発。又、外注より成果物を受入れる為の品質基準が不明確（数値基準が無い）であり、受入時のレビューが不足。

Improve

改善策	
	１．第３者によるプロセス監査・リスク監査の継続的実施。 ２．外注受入の為の品質基準の作成 及び、品質基準に基づく外注受入検査・レビューの実施。

改善効果金額（年間）　：　Ｈ／Ｓ　：　xxxx千円

Measure　＊問題の分解と優先順位

一次分解

- レビューが不足
  - **二次分解**
    - リスクの抽出が充分出来ない。（１）
    - 開発プロセスの問題抽出が充分出来ない。（２）
- 製造不具合が多い
  - 請負外注先での単体テスト不足。（３）
  - 外注受入検査が充分出来ていない。（４）

＊解決効果の大きい順に（１）～（４）

Control　＊改善を定着化するために行ったこと

プロセス監査・リスク監査を公式レビューと同様に開発プロセスの一部として定義付け、ＣＭＭにおけるＳＱＡ活動の一貫として監査を実施していく。

2．仕様変更管理表：第５章　５－１．●仕様変更管理表がもたらす効果、表５－１－１

【仕様変更管理表】

No.	管理番号	発生日	対応期限（ベースライン）	対応日	対応状況	業務名	変更内容	変更区分	修正担当	影響度範囲	難易度	リリースバージョン	対応工数（見込）	対応工数（実績）
1	PJ-001										A			
2											B			
3											C			
4														
5														
6														
7														
8														
9														
10														
合計														

3．機能モジュール進捗管理表：第５章　５－２．●モノの進捗管理、表５－２－２

業務	プログラム名称	ＥＸＥ，ＤＬＬ、ＯＣＸ等名称	開発Ｖｏｌ（Kbyte）		担当者	開発状況	コーディング（Ｃ）		単体評価（Ｕ）		結合評価（Ｋ）		不具合残件数	開発残件項目・処理日程			
			見込	実績			完了（予定日）	完了（実績日）	完了（予定日）	完了（実績日）	完了（予定日）	完了（実績日）		9月30日	10月10日	10月30日	備　考
	ＸＸＸ受信プロセス	ｘｘｘｘｘ.exe	50	60	Ａ、Ｂ	C	6/30	6/30	8/15	8/15	8/30	8/30		△△△機能			△△△後の継続処理
	ＸＸＸ送信プロセス	ｘｘｘｘｘ.exe	45	40	Ａ、Ｂ	C	6/30	6/30	8/15	8/15	8/30	8/30		△△△機能			○○○のﾊﾟﾌｫｰﾏﾝｽ向上
	通信プロセス	○○.exe	25	30	Ａ	U	6/30	6/30	7/15	7/15	7/30	7/30					
	リカバリー応答プロセス	○○RECOV.exe	35	30	Ａ、Ｂ	U	6/30	6/30	7/15	7/15	7/30	7/30			○○○処理		スケジュールのステップ分け検討
	△△受送信プロセス	△△RECV.exe	25	20	Ａ	K	5/30	5/30	6/15	6/15	6/30	6/30					
	△△送受信プロセス	△△SEND.exe	20	30	Ａ	K	5/30	5/30	6/15	6/15	6/30	6/30					
	△△起動問合せ	△△INQ.exe	30	10	Ｂ	K	5/30	5/30	6/15	6/15	6/30	6/30					
	ＦＴＰプロセス	○○FTP.exe	20	30	Ｃ	K	5/30	5/30	6/15	6/15	6/30	6/30					

最小モジュール単位までブレークダウンする

量的把握の見込み／実績を行う

担当者の明確化

工程別進捗を予定／実績日程にて行い。色彩にても直感しやすくする

品質状況の把握を行う

残件処理日程を明確にする

備考記述による問題点のフォロー

４．労働負荷状況管理表：第５章　５－２．●ヒトの消耗度管理、表５－２－５

【労働負荷状況管理表】

プジェクトメンバー氏名	先々月残業時間	先月残業時間	当月残業時間	３か月合計	１ヶ月平均残業時間	労働負荷レベル	労働負荷状況の理由	負荷軽減対策

労働負荷レベル

５：非常に過度：残業８１時間／月以上

４：過度：残業８０時間～４１時間／月

３：普通：残業４０時間～２１時間／月

２：軽度：残業２０時間～１１時間／月

１：非常に軽度：残業１０時間～０時間／月

5．プロジェクト完了報告書：第6章　6－1．プロジェクト完了報告書

●仕様変更管理の考察、表6－1－1

仕様変更管理の考察														
No.	管理番号	発生日	対応期限（ベースライン）	対応日	対応状況	業務名	変更内容	変更区分	修正担当	影響度範囲	難易度	リリースバージョン	対応工数（見込）	対応工数（実績）
1	PJ-001	xx.09.31	xx.10.31								A			
2											B			
合計														
考察	1．対応工数の見込みと実績の差異の理由および今後の改善課題 2．仕様変更管理全般についての反省													

５．プロジェクト完了報告書：第６章　６－１．プロジェクト完了報告書

●進捗管理の考察、表６－１－２

進捗管理の考察	各工程における進捗の目標と実績の差異の理由およびに今後の改善課題
要求定義	
概要設計	
詳細設計	
コーディング	
テスト	
進捗全般に関する反省	

【チェックリスト一覧表】

著者プロフィール

佐野洋（さのひろし）
現在，フリーコンサルタントとして PM ファクトリーを主宰。
数十年にわたり POS システムのファームウェア開発およびプロマネ業務に従事。
好きな言葉は「Boys be ambitious!」。
信条は「弱き者も強き者も共にその生涯を生き抜くこと」。

コンタクト

eMail：pmf_hsano@yahoo.co.jp
URL：https://pmfactory-hsano.jimdofree.com/

お問い合わせ

本書に記載されている内容についてのみのお問い合わせとさせていただきます。
またお問い合わせにつきましては、eMail：pmf_hsano@yahoo.co.jp 宛てにお願いいたします。
なお、ご質問の際には、書名、該当ページ、氏名、返信先を明記していただきますようお願いいたします。
お送りいただいたご質問には、可能な限りお答えできるように努力いたしますが、当方にて不適切だと判断されるご質問には回答を差し控えさせていただく場合もあります。あらかじめご了承のほどお願いいたします。

著作等

『SE 稼業は忘己利他（もうこりた）―現場に転がる箴言集』 技術評論社ｗｅｂサイトにて連載
URL：http://gihyo.jp/dev/serial/01/engineer-proverbs

CONTENTS