ＩＴプロジェクト

今日のチェックリスト

Today’s Checklist

ちょっと待った、その開発！　チェックリストで楽々開発

目次

■プロジェクト安全チェックリスト

Check List ＃１　開発プロジェクト安全チェックリスト　▶Check Timing：開発準備時

【見積り回答前】  **Check Timing**

□要求仕様の骨子が決定され明確に書面にて提示されていること。

□見積り回答書の内容は、要求仕様骨子の実現に見合った開発期間および開発費になっていること。

□見積り回答書には、明確な要求仕様骨子に対してのみ見積られ、不明なものについては見積りに含まれていないことを明記すること。

□見積り回答書に、開発に必要な実行条件がある場合、その条件を明記すること。

【設計着手前】  **Check Timing**

□すべての要求仕様が決定され、明確に書面にて提示されていること。

□全ての要求仕様に関して、全ての不明点および疑問点が払拭されていること。

【製造着手前】  **Check Timing**

□基本設計が完了され、基本設計書として明確に書面にて提示されていること。

□詳細設計が完了され、詳細設計書として明確に書面にて提示されていること。

□全ての設計書に関して、全ての不明点および疑問点が払拭されていること。

【単体テスト前】  **Check Timing**

□単体テスト試験書が作成済であること。

□テスト対象機能のコーディングが全て完了していること。

【結合テスト前】  **Check Timing**

□結合テスト試験書が作成済であること。

□テスト対象機能の単体テストが全て完了していること。

【総合テスト前】  **Check Timing**

□総合テスト試験書ないしは運用（シナリオ）テスト試験書が作成済であること。

□総合テスト対象機能の単体・結合テストが全て完了していること。

【成果物リーリース前】  **Check Timing**

□全ての機能が要求仕様通り、システムとして実運用を満たしていることが確認されたか。

□発見済で未修正の不具合が残存していないか。

Check List ＃2　プロジェクトを成功させるための基本条件

【開発準備時】 Check Timing

【統合管理】

□ベンダーを中心とした統合的プロジェクト管理を行うこと。

□適切なプロジェクト・マネジメント能力を発揮すること。

【コミュニケーション】

□プロジェクト傘下の全組織および全会社間の密接なコミュニケーションを行うこと。

□日次情報共有会議によるチーム内での情報共有を行うこと。

【要求仕様】

□主要な仕様を適切な時期までに凍結させること。

【リソースの確保】

□見積り交渉において、開発内容に見合った工期・予算を確保すること。

□顧客システムおよびその運用業務に精通していること。

□開発規模および難易度に応じた開発体制を構築すること。

【開発】

□顧客価値の重要度順の仕様凍結ならびに開発を行うこと。

□ドキュメントに基づいた開発を行うこと。

□データに基づく開発を行うこと。

□開発効率化・失敗の再発防止のために普段の改善活動を行うこと。

■不条理な顧客要求のチェックリスト

Check List ＃３　不条理な顧客要求への対抗策

【見積り回答前】  **Check Timing**

□開発責任者と要求側責任者の密接かつ直接的な話し合いを実行すること。

□完全な要求仕様書の早期提示時期の確約をとること。

□開発仕様はすでに凍結合意されたものに限定すること。

＊困難な要求に対しては、困難な条件の提示を行うこと。

□要求元からのＰＭ、有識技術者等の投入

□要求元からの評価テスト支援メンバーの投入

□開発要員増強のためのプレミアムコストの要求

□暫定版または優先度順のリリース、分割リリース等の条件提示を行うこと。

□「暫定版の不具合発生責任は要求者に負っていただく」と言う条件提示を行うこと。

□「暫定版の市場投入は、限定数にて実験店舗のみ」と言うような条件提示を行うこと。

□「全店稼働は正式版のみで行うこと」と言うような条件提示を行うこと。

■仕様凍結のチェックリスト

Check List ＃4　早期仕様凍結

【仕様凍結準備時】　Check Timing

□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、元請側と下請側で合意をしておくこと。
□顧客・元請・下請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。
□受注者側においても提案型仕様凍結を行うこと。
□顧客価値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。
□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。

Check List ＃5　仕様理解

【仕様調査時】　Check Timing

□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。
□要求仕様の背景や意味を必ず理解しておくこと。
□まずは仕様の全体像の把握から始めること。
□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。
□習得した内容を、ドキュメント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てはいけない。

Check List ＃6　仕様調査およびＱ＆Ａ運用

【仕様調査時】　Check Timing

□仕様内容の骨子の事前の把握と整理を済ませておくこと。
□仕様調査は、顧客価値の重要度順および基幹的仕様から始めること。
□仕様追加・変更の影響度の事前調査を済ませておくこと。
□全ての問題点・疑問点を掘り起こすこと。
□重要な仕様・基幹仕様に関するＱ＆Ａは直接対面コミュニケーションで確認すること。
□仕様内容および変更内容はチーム内（含む評価チーム）で即時的な情報共有を行うこと。
□仕様調査のＱ＆Ａ情報や変更内容は要求仕様書や設計書も同時に更新すること。
□仕様決定事項や変更事項は集中的かつ情報共有可能なシステムで管理すること。

Check List ＃7　影響度表作成のポイント

【設計時】 Check Timing

□正確な構造設計書・テーブル関連図・プロセスフローから影響度表を作成すること。

□事前の実装調査にて判明した他機能への影響情報を記録すること。

□設計作業時において知りえた影響情報を記録しておくこと。

□製造時において知りえた影響情報を記録しておくこと。

□デバッグ時に判明した他機能への影響情報を記録すること。

□単体・結合・総合の各テストにおいて判明した他機能への影響情報を記録すること。

□市場障害対応にて判明した他機能への影響情報を記録すること。

■ 要求仕様書のチェックリスト

Check List ＃8　要求仕様書のチェックポイント

【要求仕様書発行・受領時】 **Check Timing**

１．妥当であること

□顧客やユーザーのニーズと一致していること。

□上位のシステム要求仕様書などの関連する他のドキュメントとの矛盾がないこと。

□未確定項目がある場合は、どのように合意するか、依頼者と合意形成方法を決めておく。

２．あいまいでないこと

□要求仕様書に記述されている要求が、ただ一通りに解釈できること。

□要求仕様書の“良し悪し”を判断する手段や基準をもつこと。

□「範囲」を読み取れるように要求を表現すること。

□仕様は「仕様である」ことを明示し、説明は「説明である」ことを明示して記述すること。

□要求仕様書では、記述内容が“特定”できる表現になっているものを“仕様”とすること。

□要求仕様書の構成や内容は、後工程の読者に分かるように書くこと。

□「等」や「ｅｔｃ」の文言は使用しないこと。使用する場合は、○月○日までに決めるとコメントをつけること。

３．完全であること

□顧客やユーザーの、情報システムに対するニーズが漏れなく要求仕様書に記述されており、かつ図表の参照や用語の定義などの、要求仕様書の形式が整っていること。

□「境界」は早い段階で決めること。

□「要求」のモレを防ぐために、カテゴリの分類や要求の分割・階層化に漏れがない、隙間がないことを確認できるようにすること。

□要求仕様書には、「操作性」「保守性」「交換性」などの「品質要求」を記載すること。

□階層化の基準として、以下を（状況によっては組み合わせて）使い、「隙間」なく分割すること。
時系列分割（時間軸分割）／構成分割／状態分割／共通分割

□モレなく書くこと。

□要求仕様の番号をテストケースの番号とひもづけし、テストケースにモレがないことを確認すること。

□仕様をグループに分け、さらに集合を小さくし、混じり気のない仕様のグループを作る。

□＜グループ名＞に要求の性質を持たせるためには、範囲をあらわしていることを意識してグループ名を選ぶこと。

□「・・・・は、・・・・しない」という「否定表現」を避け、ｔｈｅｎとｅｌｓｅの両方を明らかにすること。

４．矛盾がないこと

□要求仕様書内部で矛盾や衝突がないこと。

□ほかの機能の仕様と衝突していることに気づくためにも、仕様は早期に展開すること。

□早い段階で全体の仕様化を行うこと。

５．重要度と安定度のランク付けがされていること

□各要求について、重要度と安定度を示す指標を明確につけておくこと。

□確認中の仕様をそのまま記述し、変わる可能性があることを明記すること。

６．検証可能であること

□開発されたソフトウェアが、要求仕様書に記述された要求を満たしているかどうかを確認可能であること。

□検査部門の人に、「検査可能」という側面から要求仕様書のレビューを実施してもらう。

□品質要求（「操作性」「保守性」「交換性」など）はテストでも確認すること。

７．変更が容易であること

□要求仕様書に対する変更が、容易に、完全に、一貫して行えるようになっていること。

□ａ）目次や索引、明確な相互参照が整備され、使いやすい構造になっていること。

□ｂ）冗長でない、つまり、同じ要求が要求仕様書内で複数個所に記述されていないこと。

□ｃ）他の要求と混ざらず、各要求を独立・分離して表現している。つまり、要求が互いに依存していないこと。

□重複なく書くこと。

□仕様書全体を「均一」に記述することにこだわらないこと。関係者間で共有できている認定仕様まで、詳細に記載しなくてもよい。

□仕様番号の確定作業は、仕様化の最初の段階では行わないこと。グループ分け確定後に行うこと。

□似た記述が続く場合に、何が違うかをすぐに読み取れるようにすること。

８．追跡可能であること

□要求仕様書に記述された個々の要求に関し、その起源が明確であり、開発が進行するに伴って作成された文書等との対応付けがとれること。

□ａ）後方追跡可能性

□ｂ）前方追跡可能性

□設計や実装の工程で明らかになった「仕様」は、要求仕様書に書き戻すこと。

□「要求」と「理由」をセットで表現すること。

□要求仕様には固有の記番号を付けること。

（参考資料：　ＩＥＥＥ８３０品質特性、ＵＳＤＭ）

■見積り問題のチェックリスト

Check List ＃９　見積り回答リスク回避

【見積り回答前】　Check Timing

□適切な開発費と開発期間の確保ができているか。

□リスク物件における分割見積り・分割開発・分割リリース等の交渉を行っているか。

□複数社体制プロジェクトにおいて、自社責任範囲を見積り回答書に明記しているか。

□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っているか。

□見積りの使用目的を確認すること。 例；顧客要求のため、受注に直結しない参考値のため、など。

□口頭での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こと。

□顧客の正式要求のものならば、要求日および妥当な回答希望日を記入した見積り依頼書を両社の正式なルートを通して、要求仕様書と共に提出していただくよう伝えること。

□正式要求書の場合、回答期限に間に合いそうもないと判断された場合は、その理由を明確にし、時間を置かず直ちに要求者と期限の交渉を行うこと。

Check List ＃１０　見積精度向上

【見積り回答前】　Check Timing

□見積り対象の仕様知識に習熟しているか。

□見積り対象システムのプログラム構造を理解しているか。

□見積り前に要求仕様の事前調査を済ませているか。

□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ったか。

□事前調査の結果をドキュメントに残したか。

□見積り対象仕様開発における過去の失敗リスクを把握しているか。

□見積り対象仕様開発に必要な技術を保有しているか。

□過去の開発案件の見積りと実績の差異の原因把握および改善を行っているか。

Check List ＃１１　見積り回答書

【見積り回答前】　Check Timing

□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。

Check List ＃１２　概算見積り

【見積り回答前】　Check Timing

□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。

□自分で見積り可能なものとそうでないものを分けること。

□未経験項目および重要リスク項目をリストアップすること。

□それぞれに対して、必要工数を大雑把に 3 段階（大・中・小）程度に分けて記入する。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。

□これらの仕事を何人で実行するかを想定しておく。

□算出した開発期間の合計を想定人数で割ったものが想定されるスケジュール期間となる。

■丸投げ問題のチェックリスト

Check List ＃１３　仕事の丸投げ状況

【見積り回答前・後】　Check Timing

□見積り回答内容以外の仕事を強要してはいないか。／されてはいないか。

□責任範疇外の仕事を強要してはいないか。／されてはいないか。

□過不足のない要求仕様書を提示しているか。／提示されているか。

□合理性・妥当性に反した開発期間・内容を強要してはいないか。／されてはいないか。

□合理性・妥当性に反した開発コストを強要してはいないか。／されてはいないか。

□仕事に関する疑問・質問に誠意をもって回答しているか。／回答を受け取っているか。

□相互の約束がお互いに履行されているか。

□優位な立場を悪用した不条理な責任転嫁が行われていないか。

Check List ＃１４　仕事の丸投げに対抗するポイント

【事前準備工程】　Check Timing

□顧客側の役割および職務責任を契約書等にて明確にすること。

□ベンダー側の役割および職務責任を契約書等にて明確にすること。

□下請け側の役割および職務責任を契約書等にて明確にすること。

□統合的プロジェクト管理により全ての関係組織・会社間の役割・責任を明確にすること。

□上司および部下の役割および職務責任を社内規定等にて明確にすること。

□自分における役割および職務責任を明確に意識すること。

□当事者同士で解決できない場合は、さらに一段上のマネジメントに相談すること。

□事前準備・改善活動等による効率化を行い、時間切迫が理由の丸投げをなくすこと。

□改善活動等によるスキルアップを行い、能力不足が理由の丸投げをなくすとこ。

□丸投げ行為は職務放棄とみなし厳罰をもって臨むこと。

□丸投げに泣き寝入りすることなく組織的な対抗措置をとること。

■相互義務履行のチェックリスト

Check List ＃１５　チームプレーにおける相互義務の履行

【開発全工程】 **Check Timing**

□顧客チームとベンダーチーム間の相互義務の履行および連携はできているか。

□同一プロジェクトに参加している他社間の相互義務の履行および連携はできているか。

□ベンダー・顧客側においては何を開発するのかを要求仕様書等にて明示すること。

□請ける側においては要求に対しての実現方法を見積り回答・設計書等で明示すること。

□要件定義チームと開発チーム間の相互義務の履行および連携はできているか。

□開発チームと評価テストチーム間の相互義務の履行および連携はできているか。

□上司と部下の間の相互義務の履行および連携はできているか。

■ベンダーリスクのチェックリスト

Check List ＃１６　基本的なベンダーリスク

【見積り時、要件定義時】 **Check Timing**

□実現不可能な納期・開発費を強制されていないか。

□複数社・複数工程間の統合的なプロジェクト管理、問題情報共有が行われているか。

□適正な要求仕様書が妥当な時期までに提示されているか。

□仕様等の不明点・疑問点についての回答時期・内容は適正か。

□ベンダー側担当者の仕様知識、技術能力、管理能力は妥当か。

□オフショア開発に対する進捗・品質管理に問題はないか。

Check List ＃１７　ベンダーリスクへの対応のポイント

【見積り・要件定義時】 **Check Timing**

□不条理な要求に対してデータに基づく交渉等により妥当な納期・コストの獲得を行うこと。

□仕様の提案や集中検討会などを通して早期の仕様凍結に全力を尽くすこと。

□要求仕様には必ず抜けや誤りがあるという前提を持つこと。

□密接なコミュニケーションを維持し、疑問点・不明点を払拭すること。

□契約外の要求に対しては追加期間・コストの要求を行うこと。

□評価時間が確保できないようなリリース直前の仕様変更要求を請けないこと。

□工程のスキップや中断などの不正な指示には従わないこと。

□開発や評価に必要な情報・機材・開発環境が適正な時期に提供されること。

□問題の解決にあたっては、一方的な丸投げを行うことなく、両者が協力して臨むこと。

□ベンダー側にて不足している役割を新たな仕事として取り込む提案ができるか。

■開発工程リスクのチェックリスト

Check List ＃１８　開発工程の３時点でのリスク回避

１．開発の入り口で押さえるリスク　　【見積り・要件定義時】 Check Timing

□適正な見積りにて妥当な開発期間と費用を獲得すること。

□あいまいな要求仕様の明確化および早期の仕様凍結に全力を注ぐこと。

□要求仕様書を常時メンテナンスし”使えるドキュメント”として残すこと。

□ＱＣＤの数値目標の設定を行うこと。

２．開発中に押さえるリスク　　【開発中】Check Timing

□あらゆる無駄の排除とリスク項目の解消を実行すること。

□設計書をリアルタイムでメンテナンスし、常にプログラム内容との同一性を保持しておくこと。

□失敗の真因・再発防止策をドキュメントとして残すこと。

□創意工夫の内容を設計書やガイドラインなどのドキュメントに残すこと。

３．開発の出口で振返ること（ラップアップミーティング）

【開発終了時】 Check Timing

□ＱＣＤの目標値と実績値を比較・分析し、問題の原因を数値データと共に明確にし、対策した結果をまとめておき、プロジェクト完了報告書にて他チームとも情報共有を行うこと。

□失敗に学ぶ。失敗あるいは創意工夫の記録を振返り、次の開発・開発者に申し送る。

Check List ＃１９　プロジェクトリスク回避

【見積り・要件定義時】 Check Timing

＊プロジェクトのリスクを認識し事前の対策を講じているか。

□見積り交渉における妥当な開発費および期間を獲得すること。

□要求仕様の早期凍結を行うこと。

□開発業務品質向上のための改善活動を実行すること。

□リーダーによる開発仕様の全体像の把握を行うこと。

□開発メンバーによる開発仕様の十分な理解を行うこと。

□開発メンバーに対する適切な仕事の配分を行うこと。

Check List ＃２０　スケジュール遅延リスク回避

【見積り・要件定義時】 Check Timing

□実行すべき内容が明確になっているか。

□開発に必要な期間・工数の見積りに間違いはないか。

□無理なスケジュールが組まれていないか。

□仕様凍結遅延ないしは仕様変更が多発していないか。

□予定目標と実績進捗は日々管理されているか。

□リーダーにおける開発内容の全体像の把握が十分か。

□担当者が仕様を十分に理解しているか。

□担当者ごとの生産性・能力・性格を考慮した仕事の配分が行われているか。

■コミュニケーション問題のチェックリスト

Check List ＃２１　コミュニケーション（基礎編）

【会議・会話の前・中・後】　Check Timing

□顧客との会議に、予想される想定問答など十分な事前準備をして臨んでいるか。

□他人の評判だけに従って自分の行動を決めてはいないか。

□発言すべきか否かを、必要性によらず感情によって決めてはいないか。

□先に相手の話を聞くなど、丁寧なコミュニケーションを行っているか。

□相手の話の途中に割り込むようなことをしてはいないか。

□毎日、短時間でもコミュニケーションを継続することで良好な人間関係を維持しているか。

Check List ＃２２　コミュニケーション（中級編）

【会議・会話の前・中・後】　Check Timing

□話をする前に、自分の中で伝えたい内容を整理しているか。

□伝えるべきこと、伝えてはいけないことをわきまえているか。

□重要な取り決めは相互の合意を文書にて残しているか。

□相手の話を良く聞かずに、自分の意見ばかりを話していないか。

□事実を伝えるよりも良く見せることに注意が傾いてはいないか。

□恥や外聞を恐れ、不明点・疑問点をその場で確認することを避けてはいないか。

□合理的かつ妥当な結論を導き出すことよりも、議論の勝ち負けにこだわってはいないか。

□客観的なドキュメントや数値データに基づいてコミュニケーションを行っているか。

Check List ＃２３　コミュニケーション（上級編）

【会議・会話の前・中・後】　Check Timing

□相手が最も聞きたいと思われることを最初に伝えているか。

□相手の能力レベルに合わせた話し方をしているか。

□仕事の指示や依頼において、相手の理解度を確認しているか。

□会話を深めるに、相手との共有体験を持つことや教養の幅を広げているか。

□会社間の重要事項に関するコミュニケーションはリーダー対リーダーで行っているか。

■会議問題のチェックリスト

Check List ＃２４　顧客との会議のポイント

【顧客会議前】　Check Timing

□開発仕様の全体像の把握、仕様の目的・意味・背景を理解しておくこと。
□事前準備・事前検討を行い、顧客からの想定質問への答えを用意しておくこと。
□顧客からの宿題事項は、状況を聞かれる前に答えること。
□自分の言いたいことよりも顧客の話を良く聞くこと。
□疑問点・不明点については臆することなく、その場で確認を取ること。
□リーダークラスにおいては、発表力やプレゼンテーション力を強化するために、普段の開発業務の中で発表やプレゼンの機会を自ら求めること。
□リーダークラスにおいては、議事進行や調整能力を磨くために、社内会議等にて議長役や書記役を自ら買って出ること。

Check List ＃２５　会議運営のポイント

【会議前】　Check Timing

□会議の目的・背景・討議内容を事前に出席者に伝えているか。
□何を討議するのか事前に決めているか。
□何を決めるのか事前に明確にしているか。
□誰が・何を・いつまでに実行するのかを決めるようにしているか。
□不必要な人を招集してはいないか。
□ダラダラ雑談に終了宣言を行うことができるか。

Check List ＃２６　日次情報共有会議のポイント

【情報共有会議前】　Check Timing

□昨日（ｏｒ今日）何を実行したかの報告を行うこと。
□今日（ｏｒ明日）何を実行するのかの予定報告を行うこと。
□今抱えている問題や困っていることを報告すること。
□メンバーは、事前に報告事項を日報に記入する習慣を持つこと。
□リーダーは、各報告に対して適時のアドバイスを行うこと。
□リーダーは、全員に共有すべき情報を伝えること。
□長時間を要する問題は当事者とリーダーで別途打ち合わせをすること。

■レビュー問題のチェックリスト

Check List ＃２７ レビューの基本

【レビュー前】 Check Timing

□重要機能順（顧客価値の順）にレビューを行っているか。

□仕様および機能の目的・意味・背景を正しく理解しているか。

□必要な仕様書や設計書および資料を用意してレビューを行っているか。

□入力条件は正しいか、処理ロジックは正しいか、出力は正しいか。

□パフォーマンス・レスポンスの考慮は適切か。

□メモリーやＣＰＵ速度等のハードウェア条件の考慮は適切か。

□想定仕様によって開発をしている部分はないか。

□過去の類似の失敗例を参考にしてレビューを行っているか。

Check List ＃２８ 効果的なレビューのポイント

【レビュー前】 Check Timing

□レビューの時間を計画的に確保しておくこと。

□レビューは対象物の全件チェックではなく基本部のチェックに絞り込むこと。

□レビュー対象物の一覧表を作成すること。

□過去のミス・不具合の傾向に基づいてレビューの重点を絞っておくこと。

□基幹機能に関する基本的な理解が正しいか。

□基幹機能の実現方法は適切か。

□中間レビューを行うこと。

□共通的なレビュー項目のひな型（チェックリスト）を作成しておくこと。

□仕様変更の影響部分のレビューも行うこと。

□担当者は、事前に自己チェックを済ませておくこと。

□レビュワーは、担当者が見落としがちな異常系、過去の評価モレ等の広い視点を持つこと。

■情緒的思考行動問題のチェックリスト

Check List ＃２９　感情本位から目的本位へ

【モチベーション低下時】　Check Timing

□自分の関心を自分自身から仕事そのものに移すこと。

□顧客の要求するものは何かということを良く理解し、その実現に向けて情熱を注ぐこと。

□たとえ嫌いな相手であっても仕事に必要なことは必ず実行すること。

□人の好き嫌いで自分の行動を変えず、誰に対しても一貫した姿勢と行動を保つこと。

□過度な自己中心性を捨て、顧客要求の実現のためにチーム／メンバーに貢献すること。

＊自分を害するものを減らしたければ、これらのことを実行する必要がある。

Check List ＃３０　感情的・情緒的思考行動の解消

【問題対応時】　Check Timing

□バグ発生に対して怒ることよりも冷静に原因追及および修復作業に努めているか。

□計画通りに物事が進まないことを自分やメンバーの性格のせいにしてはいないか。

□問題に直面した場合に合理的な判断ができているか。

　□今までの計画に想定違いや欠陥がなかったか。

　□計画を阻害するどのような新たな要因が発生したのか。

　□目の前のデータや事実から原因・理由を見つけて新たな対策を打つこと。

■目標設定問題のチェックリスト

Check List ＃３１　目標設定

【開発着手前】　Check Timing

□仕事をこなすだけで、目標の設定は不要だと思っていないか。

□現実的で達成可能なＱＣＤの目標値を設定しているか。

□目標は測定可能な数値で表されているか。

□目標は過去の実績を基準に設定されているか。

□目標達成のための具体的な手段を持っているか。

□目標値は実行手段に見合った数値になっているか。

□目標は、その変更も含めて、常時、チーム内の全員で共有されているか。

■モチベーション問題のチェックリスト

Check List ＃３２　モチベーション

【モチベーション低下時】　Check Timing

□仕事の意味を理解せずに形式的にこなすだけになってはいないか。
□単純作業にも創意工夫を働かせているか。
□意味や背景を考えて仕事をしているか。
□言われたから仕方なくやると言うような、やらされ感的な仕事になってはいないか。
□ＱＣＤの数値目標を持って、惰性に流されない仕事を行っているか。
□仕事の量や質の全体を把握することで出口（完了時期）が見えるようにしているか。
□業務中に頻繁にまたは長時間にわたって私的行為をしている者に警告しているか。
□失敗やミスの原因を明確にし、歯止め対策を講じているか。
□やる気のない者と同じ行動をしてはいないか。
□自分自身や他人を過剰にあるいは過少に評価してはいないか。
□個人に対して卑下あるいは侮辱などの行為をしてはいないか。
□他人より多いあるいは少ないという比較だけにこだわってはいないか。

Check List ＃３３　仕事への取組み姿勢

【モチベーション低下時】　Check Timing

□好き嫌いという情緒的な姿勢で仕事をしてはいないか。
□義務感を超えた誠実さをもって仕事に取り組んでいるか。
□仕事内容の意味や背景を知るために、“なぜ。”の問いかけを行っているか。
□事前準備・事前調査を行っているか。
□放置している案件がないか。
□複数の仕事を並列的に進める工夫をしているか。
□優先度を判断して仕事を進めているか。
□データやドキュメントに拠る合理性に基づいた仕事を行っているか。
□人々の納得が得られるような妥当性のある仕事を行っているか。
□残務を先送りにしてはいないか。
□顧客からの電話に居留守を使うことはないか。

Check List ＃３４　困難な仕事への対応

【モチベーション低下時】 Check Timing

□自分で対策を考えずに、他人に解決策を頼ってはいないか。

□消極的ないしは逃避的姿勢になっていないか。

□何が困難で不安なのかを具体的にリストアップしてみたか。

□事前の学習をしているか。

□事前準備をしているか。

□仲間との連携を検討しているか。

□条件付行動を選択しているか。

□オール・オア・ナッシング思考をやめ、合理的かつ妥当性のある行動を選択しているか。

□チームでリスク解消に取り組んでいるか。

□チーム内で責任分散を行っているか。

■本質把握のチェックリスト

Check List ＃３５　本質を見抜く

【問題検討時】 Check Timing

□自分の目で観察したか。

□自分の頭で考え、判断したか。

□複数の情報にあたってみたか。

□見聞を深める行動をしているか。

Check List ＃３６　思い込みの解消

【問題検討時】 Check Timing

□湧いた疑問に対して、何故。の問いかけをしているか。

□根拠となる事実や定義に従って、原典を確認した開発行動を実行しているか。

□変更の影響度を確認した上で仕様変更を行っているか。

□常識に照らし合わせた理解や判断を行っているか。

□仕様変更、日程変更等の重要情報は、原本配布／直接対話で伝えているか。

□複雑な問題に対して、複数の解決策を比較検討しているか。

Check List ＃３７　誤解を避ける

【問題検討時】 Check Timing

□文書によらず口頭のみの情報で仕事をしている場合がないか。

□数値や論理記号に乏しいあいまいな日本語記述の文書を使っていないか。

□勝手な想定ではなく、定義されたドキュメントに基づいた開発を行っているか。

□仕様の目的・意味・背景を知った上で開発を行っているか。

□全体的視野、顧客の視点で問題を考えているか。

□他者の視点で考えてみたか。

Check List ＃３８　不明な問題を解く　（数学の考え方１７か条、秋山仁）

【問題検討時】 **Check Timing**

□分類や整理をしてみたか。

□図や表にしてみたか。

□簡単な模型を作ってみたか。

□基準をそろえたか。

□数学やロジックの言葉で表してみたか。

□小さな例で試してみたか。

□難しい問題は分割して考えてみたか。

□必要条件で絞り込んでみたか。

□特定の要素に注目してみたか。

□視点を変えてみたか。

□逆に考えてみたか。

□操作は１つずつ片付けてみたか。

□知っている事実を活用してみたか。

□規則性を探してみたか。

□対応をつけて考えてみたか。

□自然からヒントを得てみたか。

□部分から全体を把握してみたか。

■優先順位問題のチェックリスト

Check List ＃３９　優先順位の設定

【作業着手前】 **Check Timing**

□仕事の意味・意図・背景を最初の時点で正確に把握すること。

□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。

□時間的余裕があるものは重要機能の順に処理すること。

□優先順位が判断できない場合は早めに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。

Check List ＃４０　仕事の優先順位認識

【作業着手前】 **Check Timing**

□ＱＣＤ（品質、コスト、納期）の優先順位は同一であることを理解しているか。

□標準的なプロセスで定義されている作業は全て必須であることを理解しているか。

　＊仕様の不明点・疑問点の事前調査や各工程のレビュー等は必須業務である。

□顧客や上司からの要求が常に第一優先とは限らないという認識を持っているか。

□自分の責任範囲を超える優先順位の変更は上司の承認を得ているか。

□要求内容の優先順位を顧客に確認しているか。

□タイムリミットが迫っているものを第一優先にしているか。

□顧客価値の重要度順で仕事を進めているか。

□緊急度・優先度の決定にあたって、要求者あるいは指示者との調整ができているか。

□中断された仕事について、再開に必要な事項のメモを残しているか。

□割り込み作業は必ず発生するという認識で、作業の前倒しを行っているか。

Check List ＃４１　割り込み作業対応

【作業着手前】  **Check Timing**

□全ての工程において、割込み作業や不測の問題は必ず発生するという認識を持つこと。

□顧客との会話を絶やさず、早い時点での割り込み要求情報をキャッチすること。

□全ての工程のスケジューリングは必ずバッファを持たせること。

□バッファを確保するために、常に業務の改善や効率化を行うこと。

□その場で内容を聞く余裕がまったくない場合は後刻の時間を指定して後で聞くこと。

□現状の仕事と割り込みの仕事についてチーム全体としての優先順位を考え、緊急度・重要度の高いものを優先させること。

□割り込みにて中断された作業については、中断情報を記録しておき再開に備えること。

□割り込ませることが不可能と判断した場合は、他の人に割り振ることや他の代替案を示すこと。

□割り込み対応が全く不可能な場合や開発工程の後半での仕様の追加および変更は、次回リリースにしていただくように交渉すること。

■ドキュメント・ベース開発問題のチェックリスト

Check List ＃４２　ドキュメント・ベース開発の遂行

【各工程開始前】　Check Timing

□基本部の要求仕様書は、見積り時に提示されること。
□詳細要求仕様書は妥当な時期までに凍結されること。
□要求仕様書に基づいて基本設計書が作成されること。
□基本設計書に基づいて詳細設計書が作成されること。
□詳細設計書に基づいてプログラム作成が行われること。
□詳細設計書に基づいて単体テストチェックリストが作成されること。
□単体テストチェックリストに基づいて単体テストが行われること。
□基本設計書に基づいて結合テストチェックリストが作成されること。
□結合テストチェックリストに基づいて結合テストが行われること。
□要求仕様書・実運用シナリオ資料に基づいて総合テストが行われること。

Check List ＃４３　ドキュメント更新

【変更・修正発生時】　Check Timing

□仕様調査時のＱ＆Ａ資料の内容は都度、設計書等に反映すること。
□仕様書・設計書等は、テスト等での不具合修正の都度、更新すること。
□仕様書・設計書等は、欠陥が発見された都度、更新すること。
□仕様書・設計書等は、派生開発の都度、更新すること。
□ソフトウェア構造設計書は、派生開発の都度、明確になった部分を更新すること。
□仕様変更影響度表は、仕様調査情報および評価テスト情報に基づき更新すること。

Check List ＃４４　ドキュメント・ベース業務の遂行

【決定事項発生時】　Check Timing

□顧客会議／連絡における依頼・指示・決定内容は議事録にて管理されているか。
□関係各社との会議／連絡における依頼・決定内容は議事録等にて管理されているか。
□社内会議／連絡における依頼・決定内容は議事録等にて管理されているか。
□顧客・ベンダーからの仕様変更は、仕様変更管理表にて管理されているか。
□依頼・指示・決定内容等は、「誰が、何を、いつまでに、どのように」が明記されているか。

Check List ＃４５　ドキュメントの見える化のポイント

【ドキュメント作成時】 **Check Timing**

□書類の品質の均一性保持のために、統一的にフォーマット化すること。

□仕様や機能の目的・背景・意味を記述すること。

□論理記号、数学的表記、図・表・フローチャート等を使用し、論理的な記述をすること。

□複雑な事象表現にマトリクス図や共通パターン表などを使用すること。

□重複を避け、シンプルな記述にすること。

□仕様変更による修正影響度表を作成すること。

□誤記、抜けおよび変更は発見・発生と同時に更新すること。

□開発ドキュメント内容とプログラムコードは、常時一対一で整合性を保つこと。

□複数の仕様書間、設計書間の矛盾した記述をなくすこと。

□異常系処理を記述すること。

□パフォーマンス・レスポンス条件を記述すること。

Check List ＃４６　業務文書作成のポイント

【業務文書作成前】 **Check Timing**

【準備】 □書く前に、相手に伝えたいことを重要なもの順にメモに書き出しておくこと。

【受け取る側の利益・心情を優先】

□最初に結論の記述から始めること。相手の要望や疑問にまず答えること。

□自分の言いたいことを優先しないこと。不始末の詫び状などでは言い訳から記述を始めるのは禁物。

【分かりやすさ】

□報告書は基本的に最初の１ページで全貌が分かるように書くこと。

□文章の主題・件名と本文の内容を合致させること。

□結論に始まり、続いてその経緯について簡略にまとめること。

どんなに長い内容でも本文は１～２ページにまとめ、詳細内容は添付資料とすること。

□文章は重要なものの順に書くこと。

□一つの文章はできるだけ短くすること。複雑な内容は複数の文章に区切って書くこと。

□箇条書き等によって簡潔な表現に努めること。

□同じ内容を、形を変えて何度も繰り返さないこと。

□重要な事とそうでないことを峻別すること。

【正確性の確保】

□「誰が、何を、いつまでに、どのように、どれくらい」等が明確に記述されているか。

□数値やデータによって質や量を表現することで、文書の客観性が保たれているか。

□あいまい表現はしない。例えば、”非常に”とか”おおよそ”などの表現は厳禁。

□文章だけで説明が難しいものは、マトリクス・図・表などを使用すること。

□技術用語は、顧客・プロジェクト・社内で統一用語を使用すること。

■基本的開発能力のチェックリスト

Check List ＃４７　ソフトウェア開発チームに必要な能力

【開発着手前】 **Check Timing**

□チームにおける日常的で密な直接コミュニケーション力

□ドキュメントの常備はもとよりちゃんと動作するソフトウェアの素早い開発力

□顧客との日々の直接コミュニケーションによる協調

□変更要求に対する俊敏な対応力

□顧客価値優先度の把握力

□意欲ある人材で構成されたプロジェクト構築力

□短期開発力

□一定のペースを持続できる開発力

□技術的な優位性の保持、良好な設計力の保持

□無駄・不要な仕事の削減力

□自律的チーム力

□定期的・効果的振り返りによる行動の変更・調節力

■緊急・危機対応のチェックリスト

Check List ＃４８　緊急対応時の行動原則

【緊急対応前】 **Check Timing**

□まずメンバーを集め、データを集めること。

□集めたデータを分析・統合しその意味を判断すること。

□対策案を決め、実行すること。

□コストよりも時間を優先すること。

□複数の緊急事項が重なった場合に時間的優先度を考慮すること。

Check List ＃４９　危機的状況からの脱出

【プロジェクト危機時】 **Check Timing**

□直ちに関係者全員を招集すること

□各人が保有する情報を全て一箇所に集めること。

□問題となる情報を整理分類し、可能な限り仕事の全体像（最終目標）を見えるようにすること。

□最終目標に対して自分たちがどの地点まで到達しているのかを想定して見ること。

□目標と乖離している残件項目を明らかにし、必要な工数の概算を見積ること。

□各担当の残件負荷のバランスを比較し、役割を再配分することで平準化を図ること。
現有勢力で対応が不可能と判断したら必要な人材と人数を投入すること。

□毎日の短時間情報共有会議を行い、各人において「①今日やったこと　②明日やる予定
③今抱えている問題」の三点につき報告をおこない、リーダーは適時アドバイスを行うこと。

■プロジェクト管理のチェックリスト

Check List ＃５０　進捗管理

【進捗管理時】　Check Timing

□プロセス管理表にて、主要イベントの時期、責任部署、提出物、受領物を管理すること。

□進捗管理表にて、主要日程および詳細業務の優先順位を考慮した管理を行うこと。

□コミュニケーション管理表にて、関係者間の打ち合わせ等が予め計画され、予測される全ての問題リスクの解消を行うこと。

□顧客提出用スケジュールと開発内部スケジュール間に矛盾や食違いがないこと。

Check List ＃５１　ＱＣＤ（品質･コスト･生産性）管理

【ＱＣＤ管理時】　Check Timing

□プロジェクト毎に、過去の品質数値と現在の品質数値を管理しているか。

□目標品質数値と実績品質数値の±差異の原因分析および対策を実行しているか。

□プロジェクト毎に見積り提示金額（工数）と実績金額（工数）を管理しているか。

□見積り額と実績値の±差額の原因分析および対策を実行しているか。

□プロジェクト毎に、過去の生産性数値と現在の生産性数値を管理しているか。

□目標生産性数値と実績生産性数値の±差異の原因分析および対策を実行しているか。

Check List ＃５２　品質の妥当性確保のポイント

【品質目標設定前】　Check Timing

□品質が根拠のある数値で表されているか。

□現在の品質数値と目標数値の設定がなされているか。

□納期優先という名目により品質が犠牲にされていないか。

□仕様変更の影響度調査を行っているか。

□開発規模に見合わないプロセス管理（ＣＭＭＩ）が強制されていないか。

　□必要最小限に絞った簡易型ＣＭＭＩプロセスの作成･運用

　□自社の経験則に基づくプロセス管理表の作成･運用

Check List ＃５３　仕事のスピード･効率

【開発着手前】　Check Timing

□事前準備を行っているか。

仕事にとりかかる前に、その仕事の内容の調査や全体像把握の実行、その仕事の段取りの計画、その仕事のリスクの把握などを実行すること。

□仕事の最中においてはＱＣＤの数値データを記録しているか。

□振り返り（ラップアップ）を実行しているか。

仕事が終了した時点で、仕事中に起こしたミス、失敗についての真因を把握し再発防止策を立て、次の仕事に備えること。

□以上のことをすべて記録しておき、ものごとを数値で語れるようにしているか。

□仕事内容を完全に把握しておらず、不明点や疑問点を放置してはいないか。

□仕事を実行するために必要な知識および能力に不足はないか。

□仕事の質および量の見積り間違いはないか。

□チーム内でのコミュニケーション不足による情報不足はないか。

■評価テストに関するチェックリスト

Check List ＃５４　単体・結合テストの効果的やり方

【単体・結合テスト前】　Check Timing

□製造の初期段階でソースレビューを実施し単体・結合テストの負担を軽くしておくこと。

□製造時のデバッグの質を上げること。

□流用可能なチェックリストは流用する。但し、内容を良く見極め単なるコピペはしないこと。

□テスト項目は必ずテスト仕様書を作成して行うこと。

□テスト実施項目の要・不要を精査すること。

□設計書の機能項目とテスト仕様書のテスト項目の“対応ひも付け”をすること。

□テスト結果のエビデンス（印刷媒体）や整理は電子的自動化ツールにて行うこと。

□自動化できるテストは全て自動化すること。

□テスト結果のレビューを必ず行うこと。

□日次会議にて不具合傾向を監視し、注力が必要な項目とそうでないものを峻別すること。

Check List ＃５５　総合テスト（システムテスト）の要件

【総合テスト前】　Check Timing

□総合テスト仕様書およびチェックリストは、要求仕様書および設計書の内容を過不足なく反映し作成されたか。

□総合テストは、総合テスト仕様書およびチェックリストに基づいて実施されたか。

□要求仕様書および設計書に規定された仕様通りに全ての機能が動作したか。

□全ての機能は、操作手順に従い、所定の入力条件に対して正しい出力が得られたか。

□実運用テスト用のシナリオ・テスト資料は作成されたか。

□シナリオ・テスト資料に基づいたテストは実行されたか。

□顧客の期待する実運用の役割通りに、システムが動作するのを確認したか。

Check List ＃５６　構成管理のポイント

【製品リリース前】　Check Timing

□人の判断や操作に依存する管理を極限まで減らすこと。

□開発中の成果物と中間リリースする成果物を管理するＰＣを分けること。フォルダを分けるだけではまだ人の判断ミスを誘発する可能性がある。

□リリース前に、作成したメディアによる実際のインストレーションおよび基本仕様の動作確認テストを行うこと。

■リーダーシップ問題のチェックリスト

Check List ＃５７ プロジェクトリーダーの役割

【開発着手前】 Check Timing

□プロジェクト全体の「ヒト、モノ、カネ、情報」の統合管理ができているか。
□開発内容の全体像を把握できているか。
□顧客要求の優先順位を把握できているか。
□適正な見積り回答が実行できているか。
□顧客との密接なコミュニケーションを通して早期仕様凍結が実行できているか。
□プロジェクト計画書が作成されているか。
□プロセス管理ができているか。
□進捗管理ができているか。
□リスク管理ができているか。
□課題管理ができているか。
□損益管理ができているか。
□プロジェクト完了報告および振り返りを行っているか。
□チーム内の日次情報共有会議を実行しているか。
□プロジェクト関係他社との情報共有・連携ができているか。

Check List ＃５８ リーダーの過重負荷軽減

【開発全工程】 Check Timing

□一人であらゆる仕事を抱え込んではいないか。
□抱えているルーチンワーク的作業を他のメンバーに分散しているか。
□ノウハウの継承等によりサブリーダーを育成しているか。
□リスク管理表などで問題の事前掘り起こしおよび解消を行っているか。

Check List ＃５９ プロセス管理

【開発着手前】 Check Timing

□プロセス管理表の作成および運用をおこなっているか。
□仕様凍結遅延防止の活動を適時に行っているか。
□妥当な設計・製造・評価工程期間の確保ができているか。
□開発各工程のスケジュールの遵守および主要イベントの進捗管理ができているか。
□各工程におけるレビューが確実に実行されているか。
□各工程における成果物の妥当性が確実に管理されているか。

Check List ＃６０　プロジェクト計画書

【開発着手前】 **Check Timing**

□プロジェクトの特徴・難易度に応じた妥当性のある開発／評価体制を構築しているか。

□開発環境の事前準備ないしは整備を行っているか。

□詳細かつ妥当な開発スケジュール表を策定しているか。

□メンバーの育成計画を考慮しているか。

□採用ルールの規定に基づいた協力会社の適正人材の投入計画を策定しているか。

Check List ＃６１　組織編制のポイント

【開発着手前】 **Check Timing**

□自律性を保てないような組織の細分化は行わないこと。

□名ばかり管理職の任命は行わないこと。

□リーダー、サブリーダー、メンバーなどのスキル階層別のピラミッド構成を築くこと。

□リーダーは複数のサブリーダーを育成し、サブリーダーはメンバーを育成すること。

□人員異動の要求に対してはチーム能力の急激な低下を防ぐために二番手のサブリーダーおよび二番手のメンバーを異動させること。

□異動させたメンバーに関しても自分の組織に必須な人材は、将来元の所属に戻す約束を取り付けておくこと。

□特定顧客に対するノウハウの属人的な保有のリスクを減らすために、必要なノウハウは必ず最新状態として技術ドキュメントに記載し更新を怠らないこと。

Check List ＃６２　人材管理

【開発着手前】 **Check Timing**

□プロジェクトは上級者から初級者に至るまで適切なスキルの人材で構成されているか。

□人材管理カードなどで個人別のスキル管理が行われているか。

□人材の基本的な条件が満たされているか。

　□必要な技術的能力を保有しているか

　□コミュニケーションに必要な自律的な報告・連絡・相談などの行動ができるか。

　□チームプレーに必要な協調性・誠実性などがあるか。

Check List ＃６３　メンタルヘルス

【開発全工程】 **Check Timing**

□メンバーを孤立無援の状態に置いてはいないか。

□メンバーの弱点を特徴として生かせる方法を模索しているか。

□他の人の一助になる行動をしているか。

□心身不調の者が支援を求めやすい雰囲気を醸成しているか。

□上位に位置する役職者は、問題が深刻化する前に発見や手当をしているか。

□仕事起因で発生した心身の障害は役職者の責任であると言う自覚を持っているか。

Check List ＃６４　プロジェクト完了報告・振り返り

【開発完了時】 **Check Timing**

□失敗の真因および再発防止策のまとめ等の振り返りが行われているか。

□ＱＣＤの目標値／実績値の対比、原因分析、対策が行われているか。

□今後の課題のまとめと振り返り結果の他チームへの横展開が行われているか。

■チームプレー問題のチェックリスト

Check List ＃６５　チームプレー問題（組織起因）

【事前準備工程】　Check Timing

□目標が設定されていない仕事が行われていないか。

□過酷なスケジュール（長時間・長期間残業）が強いられていないか。

□チーム内で相互義務の履行および相互扶助が行われているか。

□毎日の声かけや日次情報共有会議を実行しているか。

Check List ＃６６　チームプレー問題（個人起因）

【事前準備工程】　Check Timing

□やらされ意識で、仕事が行われていないか。

□繁忙時の他人からの質問等に邪見な対応をしてはいないか。

□他人を意識し過ぎて、過剰防衛的な孤立状態に陥ってはいないか。

□自分の利益に関係なく困っているメンバーの支援を行えるか。

□依頼や指示をされた仕事に対して、“無理”ですという反応ばかりしてはいないか。

□一見、困難な仕事に対して、できるための実行条件を提示しているか。

□日々の報告・連絡・相談などを行っているか。

□実行済内容、実行予定、抱えている問題等について日報に記録しているか。

□自分に可能な範囲から助け合いの行動を始めているか。

□行動できないことを自分の性格のせいにしてはいないか。

□仲間意識よりも競争相手意識が強すぎないか。

□助け合いや連帯は損なことだと思ってはいないか。

□他人に無関心ではないか。

□支援を受けることで他人に借りを作るのが嫌だと思っていないか。

Check List ＃６７　開発チームと評価チームの連携

【事前準備工程】　Check Timing

□開発チームと評価チームは開発初期の工程から連携し必要な情報を共有すること。

□開発チームは、妥当な時期に必要な部材や情報を評価チームに提供すること。

　□評価テストに耐えうるプログラムの提供

　□評価に支障を来す機能等の制限情報と対応時期情報の提供

　□単体および結合テスト成績表の提供

　□変更仕様の影響度情報の提供

　□評価テストに必要な実機環境および関連部材の提供（ベンダー側提供の場合もある）

　＊上記の開発部提供資料はリリース・ノートと呼ばれている。

■設計・製造の手抜きチェックリスト

Check List ＃６８　主な手抜き項目（設計・製造）

【設計・製造着手前】　Check Timing

□仕様の事前調査・検討をスキップしていないか。
□ドキュメントに基づく設計・製造・評価が行われているか。
□基本設計工程をスキップしていないか（基本設計書未作成）。
□詳細設計工程をスキップしていないか（詳細設計書未作成）。
□仕様実現に必要なメモリー容量、ハードウェア諸元を確認したか。
□パフォーマンス性能、レスポンス性能の考慮が抜けていないか。
□異常系処理の考慮が抜けていないか。
□各工程における自己チェックをスキップしていないか。
□各工程の内部／外部レビューをスキップしていないか。
□ソースコードやドキュメントの無節操なコピー＆ペーストが行われていないか。
□プロジェクト完了時の振り返りやラップアップをスキップしていないか。

Check List ＃６９　異常系処理

【設計着手前】　Check Timing

□各種Ｉ／Ｏ系の異常系処理は、ガイドブック等で文書化されているか。
□ＭＩＮ／ＭＡＸ制御の閾値はチェックされているか。
　□メモリーリソース
　□データ長
　□伝文長
　□カウンターサイズ
　□ポインター値
□異常系項目はチェックされているか。
　□排他制御
　□非同期制御
　□処理タイミング
　□リトライ処理
　□タイマー値
　□レジストリ
　□設定間矛盾

Check List ＃７０　プログラマーがやってはいけない手抜きの１２ヶ条

【プログラミング時】　Check Timing

□ソースコードの”コピペ”を無節操に行なっていないか。

□ベースのプログラム構造を理解しないままソースコード変更に着手していないか。

□データ処理の仕掛けを理解しないままソースコード変更に着手していないか。

□変更対象の機能の呼び出され方、関連機能との連携の仕方を理解せずソースコード変更に着手していないか。

□作られたデータはどの機能がどのタイミングでアクセスしているのか理解しないままソースコード変更に着手していないか。

□他人の書いたソースコード理解するためにそれを読むことだけしかしていないのでは。

□一つの変更要件に対して複数の担当者が関係している場合、早い段階でお互いに検討内容等の相互確認・レビューをしているか。また、みなバラバラにソース修正にとりかかり、寄せ集めたソースで一気にテストをしているのではないか。

□お互いにどこを変更したのか誰も知らないのではないか。

□一つの機能に関する多数の関数において同一巨大パラメータ構造体（２５６バイト等）を共通に使用してはいないか。

□開発中に発見された、以前から含まれていた潜在バグについていきなりソースコードの修正を行なっているのではないか。

□「プロジェクト計画書」を書いていないのではないか。

□「データ」をとっていないのではないか。

（出典；清水吉男著「派生開発を成功させるプロセス改善の技術と極意」および「要求を仕様化する技術　表現する技術」）

■評価テストの手抜きチェックリスト

Check List ＃７１　主な手抜き項目（評価テスト）

【テスト着手前】　Check Timing

□インターフェースにおける最終的な出力確認をスキップしていないか。

□テスト項目未消化のまま次工程に進んでいないか。

□単体テストをスキップしていないか。

□結合テストをスキップしていないか。

□総合テスト項目未消化のまま客先にリリースしていないか。

□発見済み未修正バグを含んだままリリースしていないか。

■手抜き防止のチェックリスト

Check List ＃７２　手抜き防止

【開発全工程】  **Check Timing**

□仕事の意味を考えて作業を行っているか。

□時間がないことを手抜きの言い訳にしていないか。

□予算不足を理由に手抜き仕事が行われてはいないか。

□合理的なプロセス管理表に基づいて開発を実行しているか。

□重要な手順抜けを防ぐためのチェックリストを運用しているか。

□手順慣れで必要なチェックを飛ばしてはいないか。

□コーディングやドキュメント作成において安易なコピー＆ペーストを行ってはいないか。

□ソフトウェアやドキュメント等の成果物の出荷前の現物確認や動作確認を行っているか。

□深刻な疲労に陥ってはいないか。

□手抜きで発生した実例を本人に示しているか。

□日次情報共有会議等で問題の共有を毎日実行しているか。

□前記問題点の改善活動をリーダー主導で行っているか。

■時間認識問題のチェックリスト

Check List ＃７３　自分の時間を確保するためのポイント

【開発全工程】 **Check Timing**

＊やるべき事は完全に実行すること。

□リスクの排除

見積り精度の向上、要求仕様の早期凍結などのリスクを事前に排除すること。

＊やる必要のないことをやらないで済むようにしているか。

□無理・無駄の排除

無理・無駄を徹底的に発見し排除すること。

＊やるべきでない事をやらないようにしているか

□責任範囲外の活動は基本的に行わないこと。

□メンバーの育成と同時に各自のレベルにふさわしい仕事を行うこと。

□部下を信頼せず、部下がやるべき仕事をやってはいないか。

Check List ＃７４　自分の時間を生み出すためのポイント

【開発全工程】 **Check Timing**

□要求仕様の早期凍結のための対策を実行しているか。

□精度の高い見積りおよびタフな交渉で妥当な納期・開発費を獲得しているか。

□顧客価値の重要度順を把握し、その順に開発・評価を実行しているか。

□失敗の原因になりそうなリスクの回避・排除を事前に実行しているか。

□問題のパターン化による対策のモレや無駄を防止しているか。

□業務効率化および無駄の排除等の改善活動を実行しているか。

□データやドキュメントに基づいた合理的な仕事の進め方を行っているか。

□柔軟かつシンプルな設計を行っているか。

□中間レビューを実行しているか。

□進捗を正確に把握するために、量だけではなく質の管理も行っているか。

□顧客間や社内における良好なコミュニケーションを実行しているか。

□求められている成果を正確に確認しておくこと。

□期限を確認しておくこと

□作業指示において、目的や意味を正しく伝えること。

□顧客との密接かつ直接的なコミュニケーションを維持すること。

□チーム内の短時間日次情報共有会議を毎日実行すること。

Check List ＃７５　時間の使い方

【作業着手前】  **Check Timing**

□事前に必要時間が許容時間内に収まる見込みをつけて仕事をしているか。

□問題の全体像の把握を先に行い、詳細な検討はその後にしているか。

□解けそうにもない問題にタイムリミットを設けて取り組んでいるか。

□解けない問題について、適時に有識者の支援を仰いでいるか。

Check List ＃７６　製品におけるパフォーマンス／レスポンス時間

【設計／テスト着手前】  **Check Timing**

□ソフトウェアのパフォーマンス時間はシステム要件を満たしているか。

□ソフトウェアのレスポンス時間はシステム要件を満たしているか。

■ノウハウ継承問題のチェックリスト

Check List ＃７７　ノウハウの継承はドキュメントによって

【ノウハウ継承前】　Check Timing

【見積り、要求仕様】

□要求仕様書品質の評価ガイドラインはあるか。

□要求仕様変更管理マニュアルはあるか。

□見積回答書作成マニュアルはあるか。

【開発管理】

□プロジェクト管理マニュアルはあるか。

□品質管理ガイドラインはあるか。

□コスト／プロフィット管理ガイドラインはあるか。

□進捗管理ガイドラインはあるか。

□生産性管理ガイドラインはあるか。

□開発／評価プロセス手順書はあるか。

【技術情報】

□統一専門用語集はあるか。

□顧客システム情報集はあるか。

□顧客システム運用情報集はあるか。

□技術情報／ノウハウ集はあるか。

□アプリケーション仕様集はあるか。

□仕様変更影響度表作成マニュアルはあるか。

□基本設計ガイドラインはあるか。

□詳細設計ガイドラインはあるか。

□プログラミングガイドラインはあるか。

□評価設計ガイドラインはあるか。

□テスト仕様書作成マニュアルはあるか。

□開発／評価ツール操作マニュアルはあるか。

□機器操作手順書はあるか。

□構成管理マニュアルはあるか。

□障害対応マニュアルはあるか。

□重要障害対応記録集はあるか。

■人材育成問題のチェックリスト

Check List ＃７８　仕事を任せる場合のポイント

【作業委任前】　Check Timing

□上司は、任せる仕事の全体像を完全に把握しておくこと（丸投げしてはいけない）。
□仕事の難易度とメンバーの能力レベルを見極めておくこと。
□作業の目的・意味・背景を説明すること。
□仕事の指示内容は口頭のみならずドキュメント資料により、漏れや誤解を防止すること。
□説明資料は、細かさも大切だが、分かりやすさに重点をおくこと。
□作業指示時に、相手の反応を確かめ、理解度をチェックすること。
□日次情報共有会議等で問題・疑問点のヒアリングや助言等を行うこと。
□部下の能力や気質を常に把握し、徐々に負荷を増していくことでその成長を図ること。
□本当の失敗を防ぐために、事前に適時のレスキューを計画しておくこと。

Check List ＃７９　社員教育の場

【教育実施前】　Check Timing

＊社員教育の場として以下を意識的に利用すること。
□チーム内における日次情報共有会議
□開発着手前における仕様の説明時
□仕様調査時における不明点および疑問点に関するＱ＆Ａのやりとり時
□レビュー時（設計、コードレビュー、単体テスト、結合テスト、総合テスト）
□不具合修正時
□プロジェクト終了時の振り返り会議

Check List ＃８０　スキルアップ

【教育実施前】　Check Timing

□チーム内の日次情報共有会議等を通して問題点などの指導を行っているか。
□ドキュメント・ベースの仕事を通してノウハウの蓄積ができているか。
□ドキュメントは常に最新状態に更新されているか。
□失敗事例集の蓄積が行われているか。
□蓄積されたノウハウは、組織内でいつでも誰でも検索・参照可能になっているか。
□改善活動を通してスキルアップを図っているか。
□学習の目標および達成時期の計画を持っているか。
□学習計画を実行しているか。
□専門分野に限らず広い領域の書籍を読んで仕事に役立てているか。

■失敗に学ぶチェックリスト

Check List ＃８１　失敗に学ぶポイント

【振り返り（ラップアップ）前】　Check Timing

□振り返り（ラップアップ）を行っているか。

□不具合の発生率や重要度のＡＢＣ分析の実行

□業務モジュール別の不具合発生率の把握

□工程別の不具合発生率・重要度の把握

□個人別、組織別弱点の把握

□再発防止対策の整理・分類および蓄積

□振り返り内容の他チームへの継承

□振り返りは、日次・週次・月次・工程毎に行われているか。

□失敗の真因および対策についてドキュメント化しているか。

後戻りや追加対応において新たに獲得した、仕様知識や実装理解について、他人でも理解できる形として全員でドキュメントに残しておくこと。

□仕様理解に全力を集中しているか。

スケジューリングミスの大元は開発仕様の理解不足による工数見積りミスであり、次回開発においては開発初期の短期間において主要メンバーにての仕様理解と妥当な見積りに全力を挙げること。

□顧客やベンダーとの密接なコミュニケーションを行っているか。

顧客やベンダーとの密接なコミュニケーションを通して、仕様ノウハウ、実装ノウハウを継続的に入手するようにすること。これらのノウハウに関しては個人レベルで保有せずに、必ずドキュメントに残してチームの財産として蓄積・利用すること。

□メンバーの育成を図っているか。

今回の開発で成長が見られたメンバーに関しては継続的に育成を図ること。

□損失の回復計画を持っているか。

次回以降の開発にて今回の超過分を取り戻すように改善活動を実行すること。

Check List ＃８２　類似不具合の発生原因と防止策

【振り返り（ラップアップ）前】　Check Timing

□不具合修正時に不具合発生の表面的な原因の修正のみしか実行しておらず、発生の真因の特定およびその恒久的な歯止めや解消を実行していないこと。

□過去の不具合を検索する仕組みや手段を持っていないこと。

□不具合情報をレビューに生かせないこと。

□過去の不具合に対する組織的な取り組みを行っていないこと。

■改善活動のチェックリスト

Check List ＃８３　改善活動への取り組み

【改善活動着手前】　Check Timing

□ＱＣＤ（品質、コスト、生産性）における問題点を把握しているか。

□自分における製品品質および業務品質を数値で把握しているか。

□組織における製品品質および業務品質を数値で把握しているか。

□自分の問題点について改善活動を行っているか。

□組織の問題点について改善活動を行っているか。

□改善活動は、当然の中核的な業務だと言う認識を持っているか。

□改善活動は、同じ失敗を繰り返さないために有効だと言う認識を持っているか。

□改善活動は、品質向上、コスト削減、工期短縮に有効だという認識を持っているか。

□それでも、「時間がないから」改善活動はできないと思ってはいないか。

■最後のチェックリスト　～誇り高きプロフェッショナルのために

□自分を信じて開発を行っているか。

私たちは、目の前で起きているさまざまな事柄について、他人の思惑を過度に忖度することなく、自分の目でしっかりと見、自分の頭で冷静に判断し、自分の意志により決定し、自分に自然に湧き起こる力で立ち向かっていきたい。

□道理に沿った開発を行っているか。

私たちは、一時の情緒や感情に流されることなく、また私利私怨に惑わされることなく、昔から連綿と続く信頼に値する道理に従い、人々の役に立つ考え方や行動をしたい。

□合理性に学んでいるか

私たちは、天然自然の運行の真理をよく表す数値・数理に基づく科学的合理性にかなった考え方や行動をしたい。

□柔軟かつ自由な発想で新たな視点を見つけているか。

私たちは、今までの自分たちの考え方に固執することなく、自分たちをとりまく変化に対し自由な心を保ち、新たな視点を見つけ、新たな取組みをしたい。

□ともに義務を果たし、共に助け合っているか。

私たちは、自分たちを取り巻く人々と共に相互の義務を果たし、相互の助け合いを行いたい。

□価値あるものを譲り合い、分ち合っているか。

私たちは、自分たちの保有する価値あるものを次の世代のものたちや劣位にある者たちに譲り合い、分ち合いたい。

□なすべきことをなし、後は運を天にまかすことができるか。

それでも物事が良くならない場合でも、あせらず時が熟すのを待ちたい。

私たちは、自分を信じ、世の道理に従い、合理性に学ぶことによって、柔軟な発想で新たな視点を見つけ、仲間とともに相互の義務を果たし、ともに助け合い、持てるものを譲り分ち合うことによって成長発展していくことを本分としたいと思う。それとも、自分を信じず他人を憎み、道理に外れ合理を学ばず、自分のみに固執し社会的な義務を果たさず、他人との競争に明け暮れ、譲り合うことも分ち合うこともしないような人間になるのか。その選択肢は常に自分の手の中にあります。

引用

イラスト：　いらすとや　http://www.irasutoya.com/