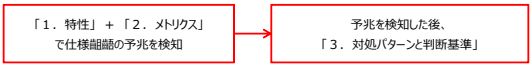
◆アンケートの目的

ウォーターフォール型の開発スタイルおいて、上流工程でお客様と十分に仕様を整合したはずなのに、開発が完了して受入れ検査を実施した際に、お客様と開発側での仕様の解釈違い（仕様齟齬）が発覚するケースがあります。そこで本研究会では、仕様齟齬の予兆を開発段階で検知するための「開発の特性」と「メトリクス」を定義するとともに、予兆を検知した後の「対処パターンと判断基準」を考えました。



今回、多種多様な開発経験・知見をお持ちの皆様に、アンケート形式で、本研究の中間成果物である「開発の特性」、「メトリクス」、「対処パターンと判断基準」をご評価いただくことで、これらの有効性を判断したいと考えております。

つきましては、ご多忙のところ恐縮ではございますが、アンケートのご回答にご協力賜りますよう、お願い申し上げます。

◆本アンケートで対象としているプロジェクトの前提条件

・上流工程での仕様確定は実施できているが、仕様齟齬の発生する可能性があるかもしれないと開発現場が感じている状態である。
・Q&Aだけでは、なかなか解決しない問題であり、Q&A以外の効果的な確認方法が必要である。
・想定する開発プロジェクト
開発期間：半年～1年
開発規模：最大時　10人／月以上の工数
開発対象：エンタープライズ
開発プロセス：ウォーターフォール型
開発形態：受託開発

１．仕様齟齬の予兆の判断で考慮すべき「開発の特性」

仕様齟齬の発生に関係がありそうな「開発の特性」

用途：各開発工程の中間および完了時に、以下の「開発の特性」と「メトリクス」の情報を収集して仕様齟齬の予兆を判断する。

No.	対象	開発の特性	用途
C1	製品	製品の新規性（新規／保守／継続）	・予兆の判断材料として
C2		技術の新規性（既存技術のみ／新規技術あり）	・予兆の判断材料として ・予兆を察知した後で、確認の優先順位を決めるため
C3		顧客内で仕様についての意見が分かれていた機能があるか？	・予兆の判断材料として
C4	顧客	経験・実績	・予兆の判断材料として
C5		新規顧客／継続顧客	・予兆の判断材料として
C6		性格（意見をコロコロ変える等）	・予兆の判断材料として
C7		開発内容の理解度	・予兆の判断材料として ・予兆を察知した後で、対処方法パターンを決めるため
C8		開発への関与度、協力度	・予兆の判断材料として ・予兆を察知した後で、対処方法パターンを決めるため
C9		開発途中での担当者の交代の有無	・予兆の判断材料として
C10		窓口は一本化されているか	・予兆の判断材料として
C11	開発チーム	経験・実績	・予兆の判断材料として
C12		性格（先入観が強い、慎重　等）	・予兆の判断材料として
C13		業界知識・製品ドメイン知識	・予兆の判断材料として
C14		開発途中での担当者の交代	・予兆の判断材料として
		以下は、予兆を察知した後のサポート情報として利用	
C15	製品	コア機能はどれか	・予兆を察知した後で、確認の優先順位を決めるため
C16		仕様が複雑な機能はどれか	・予兆を察知した後で、確認の優先順位を決めるため
C17		顧客が特に気にしている機能・フィーチャーはどれか	・予兆を察知した後で、確認の優先順位を決めるため

２．仕様齟齬の予兆を検知する際に利用する「メトリクス」

用途：各開発工程の中間および完了時に、以下の「メトリクス」と「開発の特性」の情報を収集して仕様齟齬の予兆を判断する。

No.	測定単位	メトリクス※注１	予兆ありと判断する方法
M1	予兆検知の単位※注２	前の工程の内容だったQ&Aの数（多さ）	（前の工程の内容のQ&Aの数／（仕様書のページ数、または見積もり規模））が多い
M2		仕様変更回数（多さ）	（仕様変更の数／（仕様書のページ数、または見積もり規模））が多い
M3		Q&Aの数の変化（増加量大）	前工程でのQ&A回数に対する、現工程でのQ&A回数の増加量が大きい
M4		Q&Aの数（少なさ）	【前提条件1】または【前提条件2】が成り立ち、且つ （Q&Aの数／（仕様書のページ数、または見積もり規模）） が小さい →【前提条件1】新規機能がある、または新技術を使っている →【前提条件2】前工程の工数が予定工数より極端に多い、または少ない（もっともな理由があれば問題なし）
M5		関連するQ&Aの数（多さ）	一つのQ&Aが発端となり、複数のQ&Aが発生するケースが多い
M6		仕様の確定が遅れた機能か（有無）	要求定義完了時に仕様が確定しなかった箇所であるか
M7		開発組織内レビューまたは顧客レビューの 工数・時間（少なさ、短さ）	【前提条件1】が成り立ち、且つ レビュー工数より少ない、または時間が短い（もっともな理由があれば問題なし） →【前提条件1】新規機能がある、または新技術を使っている
M8		Q&Aの数に対する仕様変更回数（少なさ）	（Q&Aの数／（仕様書のページ数、または見積もり規模））に比べ仕様変更回数が少ない
M9	プロジェクト全体	プロジェクト全体での仕様変更回数（多さ）	プロジェクト全体での（仕様変更の数／（仕様書のページ数、または見積り規模））が多い

※注１：文中の多少や大小などについてのより具体的な基準については、今後の課題としたい。
本アンケートでは、何等かの基準から多い、少ないと判断できた場合、そのメトリクスが仕様齟齬の予兆を検知することに有効であるか、取得可能であるかという視点から回答いただきたい。
※注２：サブシステム単位で検知したい場合は、サブシステム毎にQ&Aの数や仕様書のページ数などのメトリクスを取得する。
業務機能単位で検知したい場合は、業務機能毎にメトリクスを取得する。

3.仕様齟齬予兆検知後の「対処パターンと判断基準」

用途：仕様齟齬の予兆を検知した後、以下の表を使って対処パターンを決める。

表中の英字表記は、判断基準に対する各対処パターンの中での優位順（A:最も良い→D：最も悪い）を示している。

			対処パターン				
No.	分類	判断基準	①顧客に詳細設計書を**再度レビュー**をしてもらう	②**テストケースやテストシナリオ**を作成して顧客に確認する	③気になる機能のみをスケジュール前倒しで**実装（完成版）**をおこない顧客にデモ実施	④気になる機能のみをスケジュール前倒しで**プロトタイプ（育てて最終的に使うもの）**作成し顧客にデモ実施	⑤気になる機能のみをスケジュール前倒しで**モックアップ（最終的に捨てるもの）**を作成し顧客にデモ実施
J1	C:コスト	仕様齟齬を確認するためのモノを作る作業のコストの安さ	–	A	B	D	C
J2	Q:品質	齟齬の発見のしやすさ ＝「顧客に見せるモノ」の精度の高さ	D	A	A	B	C
J3	D:納期	顧客が確認できる状態になるまでの期間の短さ	–	A	D	C	B
J4	変更難度	顧客側にかかる作業負荷の軽さ	–	D	A	C	A
J5	変更難度	開発側に必要な技術・ノウハウの難度の低さ	–	A	A	D	A
J6	変更難度	ウォーターフォール型開発からの移行のし易さ	–	A	A	D	A
J7	変更難度	他の工程、全体スケジュールに与える影響度の低さ	–	A	C	D	B
J8	変更難度	顧客との調整（協力の引き出し易さ、顧客の時間の融通の）のし易さ	D	A	A	B	A
J9	C:コスト	仕様齟齬が見つかった場合の無駄の少なさ	–	A	D	C	B

アンケート0.一般質問

あたなの業務での役割や経験についてお伺いします。

Q 1 現在の役職は？

担当者	リーダ職	管理職	役員	その他
7	**16**	**21**	**0**	・基本設計フェーズで30人/月 程度のシステムのサブシステムリーダを担当 ・組織のラインマネージャーとして、各種開発PJTの責任を担う立場（PMは別の担当者がいる。）

Q 2 ソフトウェア開発経験はどのくらいお待ちですか？

5年未満	5年以上 10年未満	10年以上 15年未満	15年以上	その他
3	**8**	**13**	**19**	・4年8か月

Q 3 マネジメント（プロジェクト管理）経験はどのくらいお持ちですか？

5年未満	5年以上 10年未満	10年以上 15年未満	15年以上	その他
15	**15**	**13**	**0**	

Q 4 主に携わっているソフトウェア開発の分野はどちらに該当しますか？

エンタープライズ系	組込み系
40	**5**

Q 5 どのような開発プロセスを経験したことがありますか？（複数回答可）

ウォーターフォール型	スパイラルモデル	プロトタイプモデル	スクラム	その他
42	**7**	**11**	**9**	・ラップオーバー

Q 6 開発の初期段階でお客様と十分に仕様を整合したはずだが、開発の最終成果物を納品した際に、お客様から意向と違っている（仕様齟齬）と言われた経験はありますか？

経験がある	経験がない
36	**7**

アンケート1.仕様齟齬の予兆の判断で考慮すべき「開発の特性」

SQIP第一分科会で討議している"仕様齟齬の予兆の判断で考慮すべき「開発の特性」"について、ご意見をお聞かせください。
（"1.仕様齟齬の予兆の判断で考慮すべき「開発の特性」"を参照してください。）
（選択式の項目は、解答欄のセルをクリックし、ドロップダウンリストから●を選択してください。）

予兆の判断で考慮すべき「開発の特性」について

Q 1 以下の各特性は、仕様齟齬の予兆の判断で考慮すべきだと思いますか？

No.	対象	開発の特性	非常に考慮すべき	ある程度考慮すべき	あまり考慮すべきでない	全く考慮すべきでない	理由 （※任意："あまり考慮すべきでない"または、"全く考慮すべきでない"を選択した場合、その理由を記載してください。）
C1	製品	製品の新規性（新規／保守／継続）	**26**	**16**	**1**	**0**	・新規でも保守であっても仕様の齟齬は起きえるので新規性の関係性は低いと考える ・新規の場合、仕様が完全にFIXできないところが発生する
C2	製品	技術の新規性（既存技術のみ／新規技術あり）	**15**	**22**	**6**	**0**	・新規技術の機能・性能・制約などの技術齟齬はあるが、顧客との仕様齟齬は少ないと考えているため ・技術は実現するための手段であり、仕様齟齬との関係性は低と考える ・仕様確認の結果、技術を選定するという順番なら考慮不要 ・新規技術であって振る舞いに齟齬がないようにすれば 仕様齟齬は発生しないと考える。 ・利用技術は実現可否に対する影響であって仕様には影響しないと思います ・新規技術の場合、仕様通りに作りこめるか不確定要素がある
C3	製品	顧客内で仕様についての意見が分かれていた機能があるか？	**29**	**12**	**2**	**0**	・意見の分かれ方というより、あるべきとの比較で考えるべき。あるべきの認識違いという観点なら考慮は必要となる ・仕様が顧客先でFIXしていに可能性
C4	顧客	経験・実績	**20**	**21**	**2**	**0**	・仕様の決定に関係が薄いと思われるため ・顧客の仕様の理解度が変わる可能性
C5	顧客	新規顧客／継続顧客	**18**	**20**	**5**	**0**	・仕様の決定に関係が薄いと思われるため ・新規・継続で分かれるのは顧客の特性が把握できているか否かだが、特性については他の問いの観点に包含されるので、それ以外で考慮すべき事項は特にない ・継続顧客で慣れが出ると「なあなあ」になりふとしたことから齟齬が生まれる。新規/継続でアプローチを変えるべきではない。 ・新規かどうより担当者のスキルが大きいと思われます ・新規かどうよりどのような顧客なのかが重要だと思うので、他の特性でカバーできると判断。 ・顧客の仕様の理解が変わる可能性
C6	顧客	性格（意見をコロコロ変える等）	**19**	**20**	**4**	**0**	・仕様の決定に関係が薄いと思われるため ・最終的にFixした仕様を、エビデンスに落とした上で 合意しておけばよいと考えるため ・仕様を（誰でも間違えない）明確な定義がされていれば、問題ない。顧客の性格でFIXした仕様を変えるのはコンプライアンス違反
C7	顧客	開発内容の理解度	**24**	**18**	**1**	**0**	・開発内容を理解していても、顧客が見るのは成果物なので。 ・顧客の仕様の理解が変わる可能性
C8	顧客	開発への関与度、協力度	**19**	**20**	**3**	**1**	・正しく仕様Fixされていれば、開発工程にはいってからの関与度、協力度はあまり関係ないと考えるため ・開発内容を理解していても、顧客が見るのは成果物なので。 また、開発に協力する顧客が利用ユーザでないことも多いため。 ・本特性と仕様変更は関係ないと考える
C9	顧客	開発途中での担当者の交代の有無	**25**	**17**	**1**	**0**	・開発内容を理解していても、顧客が見るのは成果物なので。 ・齟齬ではないが、仕様がかわってしまう可能性がある ・顧客の仕様の理解が変わる可能性
C10	顧客	窓口は一本化されているか	**16**	**23**	**4**	**0**	・一本化は良いと思うが、窓口担当に依存するだけで、キーマンを抑えているかが重要 ・窓口が一本化されていても、その先がどのように伝わっているかはわからないため。 ・要件確定の大変さにはかかわるものの認識齟齬とは関連がなさそうに思います。 ・窓口が一本化されていなくても要件が関係者全体で整合されていれば問題ない。 ・顧客の仕様の理解が変わる可能性
C11	開発チーム	経験・実績	**18**	**24**	**1**	**0**	・仕様の決定に関係が薄いと思われるため ・FIXした仕様通りのものを作れないのは問題外
C12	開発チーム	性格（先入観が強い、慎重　等）	**9**	**29**	**3**	**2**	・仕様の決定に関係が薄いと思われるため ・正しく性格を把握できていることが保証できないため ・性格のみで判断する必要は感じない。どのような性格の人材でも実績を残しているのであれば、マイナスの性格を補う能力を持っていると判断している。 ・一人一人の性格をどう調べるか？調べてもそれが正しいのかが不明。 ・FIXした仕様通りのものを作れないのは問題外
C13	開発チーム	業界知識・製品ドメイン知識	**21**	**20**	**2**	**0**	・既存知識は、良い面と悪い面が存在する。未知業界・製品でも経験・実績があれば対応可能だと考える。 ・要件の仕様についてはある程度限定されるため、業界知識はあればなおよいが、考慮はマストではない認識 ・「経験・実績」との違いは？「ソフト開発の経験はあるが、対象製品や業界の知識（経験）はない」を想定している？ ・FIXした仕様通りのものを作れないのは問題外
C14	開発チーム	開発途中での担当者の交代	**16**	**22**	**5**	**0**	・仕様の決定に関係が薄いと思われるため ・リーダのコントロールの問題だと思う ・リーダクラスの交代であれば考慮すべきだが、仕様はリーダが抑えている前提のためメンバ入替は特に考慮不要の認識 ・仕様の理解ができていない可能性

その他、仕様齟齬の予兆の判断で考慮すべき特性がありましたら、記載してください。

・開発チーム内に有識者・権限者が複数存在する。
・コア機能でないが仕様が複雑な機能、またはコア機能でない横断機能については、上流フェーズで詳細な検討がなされないために祖語が生じるケースがある。
・削除機能など（ウォークスルーなどで優先度が低いためにケースに盛り込まれないことがある）
・① 要件定義プロセス ： お客さまの真の要件を引き出し、コミットされているか。 （レビュー、検収、ドキュメント）
② 仕様決定プロセス ： お客さまの業務の変更点が認識されているか。 （レビューと検収、ドキュメント）
③ お客さま体制 ： 開発側から見た担当者が、お客さま社内で決定権を持っているか、あるいは適切に権限者の承認を得ているか。
④ お客さまの業界の変化スピード ： 開発途中で常識・制度が変わる場合
・場合によっては、顧客が丸投げのケース（結果だけ確認）があるため、必ず仕様整理を問う場合は選択型でのメリット・デメリットを伝え（対処方は聞かないようにする）判断するよう心掛けている。
・プロジェクト憲章や企画書の妥当性
仕様の承認プロセスの有無
・【顧客】
・所属部門（ユーザなのか、情シスなのか、決裁権限はあるのか？）
・顧客側の受入れ体制（依頼部門以外が関与するか？品管、営業etc）
・見積時に迷う点、スコープを落としてコストカットを合意の上プロジェクトを開始した場合に、仕様齟齬が発生する傾向がある。「仕様カットしたがここは入っている認識だった」などの顧客見解と、開発側の認識齟齬。
・カウンター、ステークホルダー等の顧客側の体制がどうなっているか
開発チームから仕様決定者までの距離感が遠いと、伝言ゲームで齟齬や漏れが発生する。
・顧客の組織構造も考慮すべき。（窓口とエンドユーザが遠い。窓口と上司の関係性など。）
例）エンドユーザにヒアリングした結果、やっぱり違う回答がきた。上司に承認を取ろうとして結果、否認された、など
あとあとでひっくり返される可能性が多分にある。
・要件の内容：実現したい事やその理由（要件）と実現方法（仕様）が明確に区分けされているか
・顧客側の担当者と、実利用者との関係。（実利用者の意見が要件に含まれているか）
利用目的、利用者、利用状況が明確になっているか。変わっていないか。
・顧客の仕様FIXの意思。"絶対変更無し＆完璧"、"もしかすると変わるかもしれない、（デザインなど）マッチしないかもしれない"などのあいまいさがあるか。
・エンタープライズ系の開発では、基幹システムになると仕様の量が膨大（１機能だけで数百ページになる事もあり、それが20機能とかあると・・・）となるため、１人で全ての仕様を把握するのは困難となるため、
開発要員全員で、ドメイン知識等をカバーできるようにしておかなければいけない。
・他システムとの通信が多いため、特にその他システム間との折衝が出てくる場合は、他システム間での通信仕様に齟齬が出ないよう、相手システムの挙動はある程度理解しておく必要がある。

予兆を察知した後で、確認の優先順位を決めるための「開発の特性」について

Q 2 以下の各特性は、予兆を察知した後で、状況確認の優先順位を決めるために必要だと思いますか？

No.	対象	開発の特性	非常に必要である	ある程度必要である	あまり必要でない	全く必要でない	理由 （※任意："あまり必要でない"または、"全く必要でない"を選択した場合、その理由を記載してください。）
C2	製品	技術の新規性（既存技術のみ／新規技術あり）	**10**	**23**	**7**	**1**	・お客さまの業務にとって、裏でうごいている技術はブラックボックスでよいことが多いため。 ・仕様齟齬との関係性は低いと考える ・新規技術であって振る舞いに齟齬がないようにすれば 仕様齟齬は発生しないと考える。 ・利用技術は実現可否に対する影響であって仕様には影響しないと思います
C15	製品	コア機能はどれか	**30**	**10**	**1**	**0**	・"コア機能"＝"この製品の勘所"なら優先度を上げる必要あり。
C16	製品	仕様が複雑な機能はどれか	**24**	**17**	**0**	**0**	
C17	製品	顧客が特に気にしている機能・フィーチャーはどれか	**20**	**20**	**1**	**0**	・C17については、考慮が必要だが顧客が気にしている分、顧客から情報が出やすいため他の項目にくらべ優先度は低い印象 ・"顧客が気にしている機能"＝"この製品の勘所"なら優先度を上げる必要あり。

その他、予兆を察知した後で、確認の優先順位を決めるために必要な特性があれば、記載してください。

・多機能への影響度の大きさ。
・C2,C16等に関連するが、見積もり時の工数が大きさ。
・横断機能等、担当領域があいまいな機能であるか。
・コア機能に含まれるかもしれませんが、仕様齟齬だった場合の影響範囲の大きい機能はどれか？
・開発案件でインパクトが大きい箇所から優先度をつけていく。仕様・認識齟齬が発生し場合は、リカバリープランを提示（対策の優先度つけ）し、対応方針を双方理解頂くことが重要。また、後戻りが発生しているという状況説明をメンバーに周知することも重要（計画外の作業工数が発生していることをメンバーに認識してもらうこと）
・他業務、システム、他機能への影響度確認作業に必要なリードタイム
・【顧客】
・当初予定のリリースタイミングから、一部延期してもよい機能があるか？（段階リリースを許容できるか？）
・業務インパクト
リリースなどの開発計画に大いに関係するために早急に確認して対応する必要がある。
・顧客が気にしている機能は、顧客も慎重に取り組んでいる部分なので逆に安心。
コアな機能で齟齬が発生すると致命的なので、コアな機能に注力したい。
・まずは、要件が変わっていないか確認するため、C3「顧客内で仕様についての意見が分かれていた機能があるか？」やC6「性格（意見をコロコロ変える等）」、C9「開発途中での担当者の交代の有無」を問合せて、何かしら手を打ってみてはどうでしょうか。
・仕様変更に陥った場合に開発リスク（開発影響度高）が高いものは優先的に対処すべき
・開発するシステムによるかもしれないが、最も重要なのは、「競合する機能はどれか？」を把握しておく事だと思います。
これまでの経験で最も発生していたのが、「同期実行」、「非同期実行」の双方が、同じデータベースの同一テーブルにアクセスする場合でした。 排他漏れでデットロックを起こす可能性が高まります。
（例）これはもっともポピュラーな例ですが・・・、仕様を明確にして対応しないとマズイです。
・SUICA等で、右の事が同時に発生するケース。 ⇒お金をチャージする。 ⇒改札でお金を消費する。 ⇒職員が端末でお金を登録する。 （左2つは非同期でQUEから実行？？、右はクライアント端末との同期）
・銀行等で、右の事が同時に発生するケース。 ⇒お金を入金する。 ⇒お金を引き出す。 ⇒職員が端末でお金の引き出し作業を行なう。 （左2つは非同期でQUEから実行？？、右はクライアント端末との同期）

予兆を察知した後で、対処方法（パターン①～④）を決めるための「開発の特性」について

Q３ 以下の各特性は、予兆を察知した後で、対処パターン①～⑤（Sheet："3.仕様齟齬予兆検知後の対処パターンと判断基準"）を選択するために必要だと思いますか？

No.	対象	開発の特性	非常に必要である	ある程度必要である	あまり必要でない	全く必要でない	理由（※任意："あまり必要でない"または、"全く必要でない"を選択した場合、その理由を記載してください。）
C7	顧客	開発内容の理解度	**20**	**20**	**1**	**0**	・顧客／開発ベンダーの関係性は対立構造ではなく、協業関係とすべきという前提の上、相互に専門領域については任せる形が望ましいと考えている。（会話できる程度の知識があれば問題ない）
C8		開発への関与度、協力度	**22**	**16**	**3**	**0**	・開発工程について理解していることが重要と考える。関与や協力をしていても理解してもらえなければ意味がない。 ・本特性と仕様変更は関係ないと考える

その他、予兆を察知した後で、対処パターン①～⑤（Sheet："3.仕様齟齬予兆検知後の対処パターンと判断基準"）を選択するために必要な特性があれば、記載してください。

・C8と同様だが顧客の主要業務の忙しさ（打ち合わせ時間が取れないなど）。
・顧客の性格（意見をコロコロ変える等）
・顧客の組織構造も考慮すべき。（窓口とエンドユーザが遠い。窓口と上司の関係性など。）
例）エンドユーザにヒアリングした結果、やっぱり違う回答がきた。上司に承認を取ろうとして結果、否認された、など
あとあとでひっくり返される可能性が多分にある。
・顧客の仕様FIXの意思。"絶対変更無し＆完璧"、"もしかすると変わるかもしれない、（デザインなど）マッチしないかもしれない"などのあいまいさがあったか。

その他（フリーコメント）

その他、ご意見等ありましたら、記載をお願いします。

※ご協力ありがとうございました。

アンケート2.仕様齟齬の予兆を検知する際に利用する「メトリクス」

SQiP第一分科会で討議している"仕様齟齬の予兆を検知する際に利用する「メトリクス」"について、ご意見をお聞かせください。
（"２．仕様齟齬の予兆を検知する際に利用する「メトリクス」"を参照してください。）

メトリクスの有効性について

Q１ 仕様齟齬の予兆の検知における、以下の各メトリクスの有効性についてお答えください。

No.	測定単位	メトリクス	非常に有効である	ある程度有効	あまり有効でない	全く有効でない	理由（※任意："あまり有効すべきでない"または、"全く有効でない"を選択した場合、その理由を記載してください。）
M1	予兆検知の単位	前の工程の内容だったQ&Aの数（多さ）	**8**	**27**	**8**	**0**	・取得の実現性が低いため。 ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・難易度や顧客の要求定義の質に依存するから ・内容にもよるが、QAが多いほうが顧客の仕様への理解が深い ・規模・複雑さ・担当者のレベル等の要素もあり一概には判断できないかと思います ・仕様分析モレが見つかり質問できているのであれば問題ないように思う。齟齬と相関があるのか不明。
M2		仕様変更回数（多さ）	**26**	**16**	**1**	**0**	・難易度や顧客の要求定義の質に依存するから ・仕様変更した時点で齟齬が解消されており、その後の発生率は下がるのでは。
M3		Q&Aの数の変化（増加量大）	**6**	**26**	**8**	**3**	・取得の実現性が低いため。 ・仕様齟齬の発生に関係が薄いと思われるため ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・質問の意味がよく分かりません。何を起点として増加量を判断するのかがわからない。 ・顧客の関与度によってQAの数は変動する可能性があり、QAの数そのものでは仕様齟齬の予兆検知は判断できない M1はQAの内容が前工程のものなので、内容で判断可能 ・工程による気がするため ・フェーズ毎にQAが増加するか減少するかを測定する方がよいと考える。 ・工程によるためどのタイミングかの見極めが必要と思われます
M4		Q&Aの数（少なさ）	**15**	**21**	**7**	**0**	・仕様齟齬の発生に関係が薄いと思われるため ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・難易度や顧客の要求定義の質に依存するから ・顧客の関与度によってQAの数は変動する可能性があり、QAの数そのものでは仕様齟齬の予兆検知は判断できない M1はQAの内容が前工程のものなので、内容で判断可能 ・シンプルな機能、マスタメンテナンス等はQAとしては少なくなるため、機能による ・規模・複雑さ・担当者のレベル等の要素もあり一概には判断できないかと思います ・数が少ない＝仕様の詰めが甘いとは限らない
M5		関連するQ&Aの数（多さ）	**10**	**26**	**6**	**1**	・仕様齟齬の発生に関係が薄いと思われるため ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・顧客の関与度によってQAの数は変動する可能性があり、QAの数そのものでは仕様齟齬の予兆検知は判断できない M1はQAの内容が前工程のものなので、内容で判断可能 ・シンプルな機能、マスタメンテナンス等はQAとしては少なくなるため、機能による ・規模・複雑さ・担当者のレベル等の要素もあり一概には判断できないかと思います
M6		仕様の確定が遅れた機能か（有無）	**7**	**25**	**10**	**1**	・仕様確定の優先度が低いだけならば問題ないと判断する。 ・仕様確定が遅れたからと言って必ずしも仕様齟齬の予知を検知するのに有効かというとそうでもないから。 逆にきちんと仕様を決めたからこそ遅れた可能性もあるため ・仕様を深く詰めた結果、確定が遅れるケースがあり、この軸のみで判断はできない。 ・左記の機能数が多い場合は有効と考える。 ・その後の開発期間の調整度合いによる ・遅れても十分に議論されて確定した機能であれば、仕様齟齬を含む可能性は少ない ・明確であり、変更がなければ問題ない。変更があるならM2で対応できるのではないかと思う。
M7		開発組織内レビューまたは顧客レビューの回数・時間（少なさ、短さ）	**20**	**18**	**4**	**1**	・仕様齟齬の発生に関係が薄いと思われるため ・長い、短いの判断ができないと思う ・計測データにバラつきがでるため ・仕様の理解が十分できたことが重要。時間ではない ・これまで、レビューによる仕様分析による齟齬はかなり多いので、必要だと思います
M8		Q&Aの数に対する仕様変更回数（少なさ）	**6**	**23**	**9**	**4**	・仕様齟齬の発生に関係が薄いと思われるため ・複雑な機能であるがゆえに仕様書の記述量が多いが、要求として練りこまれている場合もあると思われる ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・関連性がよくわかりません 十分な確認がとれていれば、仕変も少ない理解 ・機能の複雑さ・ドキュメントの質・作業者・顧客担当等、さまざま内容によって変動するため、判断できない。 ・基本的にQAそのものでコストは発生しないので、顧客の特性、関与度等によって質も量も変わるものに対し、仕様変更はコスト面も含めた一定の判断が入る為、ある程度判断基準としては一定の傾向が出ると思う（要件適合度や業務負荷等） この為、一般的な統計としてQAの数と仕様変更の回数に直接の関連性は無いと思われる ・Q&Aの数と仕様変更回数は比例しないと考える ・QAで解決した内容に仕様齟齬がある可能性が高いと考える。 ・判断できませんでした。 ・仕様変更が少ない＝完成度が高いとも取れる
M9	プロジェクト全体	プロジェクト全体での仕様変更回数（多さ）	**19**	**23**	**1**	**0**	・フェーズが後半であれば、重要度が増すと思う。

その他、仕様齟齬の予兆の検知に有効なメトリクスがありましたら、記載してください。

・M6に関連し、（要件定義の工数/仕様書の枚数）が少ない場合などは考慮する必要があると思われる。
・仕様変更時における他サブシステムなどへの影響調査に掛ける工数・時間（少なさ、短さ）
・① 新規機能数 ： お客さまが経験したことがなく、システム稼動後のイメージが湧いていない可能性があるため。
・基本設計書などの上流工程における資料数：少（別紙なども含む）ケースもあいまいな検討の場合が高いため、下流工程のキックオフ時にはリスク管理としてあげるよう注意しておく。その分のリスク値を見積もり段階で組み込むこと。
・あるべき仕様変更(工程の途中で発生してよい仕様変更) ／ 仕様変更数 →少ないほど危険。
→前工程で確定すべき内容が後で発覚したものを仕様変更とした率で判断する
・テスト仕様書の顧客レビューにおける指摘の少なさが認識齟齬が発生する傾向が強い。開発側で作成した試験シナリオ（主にST工程）について、顧客が確認時間を確保できず十分なレビューができていない状態は受入時に認識齟齬が発生する傾向にある。
・Q&Aのやり取り回数
※1つのQ&Aで何度もやり取りを繰り返しているケース
・内部と外部QA数のギャップ（内部のQA数が多いが外部のQAが非常に少ない、など）
・仕様変更による、影響範囲(広さ)
・仕様の規模が大きいと、1つの変更が大きな影響を与える事があるため（1つの変更が、他の機能にブッキングしていて、そちらも変更が必要になるケースがあるため）、構成管理ツールとISSUE管理ツールを連携させて、仕様の変更結果が構成管理ツールに登録されたと同時に、プロジェクトメンバー全員に確実に通知されるような仕組みが必要だと思います。 （だいたいやっている？？）

メトリクスの実現性について

Q 2 以下の各メトリクス取得の実現性についてお答えください。

No.	対象	開発の特性	容易に実現できる	実現できる	実現が困難	実現不可能	理由 （※任意："あまり有効すべきでない"または、"全く有効でない"を選択した場合、その理由を記載してください。）
M1	予兆検知の単位	前の工程の内容だったQ&Aの数（多さ）	**15**	**20**	**7**	**0**	・前の工程のQ&Aと識別できるかどうか疑問である。 ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・各Q&Aに対する評価をする必要があるため、実現はできると思うが、容易ではない ・前工程だったかどうかの判断基準が難しい ・MTGや口頭でのやりとりまで踏まえて正確な数字を取得することが困難 ・どの工程のものか判断しにくい ・何をもって多いとするか、少ないとするか、またその粒度によっても異なる、何を持って遅いとするかなど、定量評価が難しそう
M2		仕様変更回数（多さ）	**27**	**14**	**1**	**0**	・上流フェーズの場合、仕様変更とレビュー指摘の切り分けが難しい（変更管理として記録していない場合が多い） ・正確な回数の把握は工夫が必要かもしれない
M3		Q&Aの数の変化（増加量大）	**19**	**16**	**6**	**1**	・変化を見るのであれば、件数だけでなく、Q&Aの内容も考慮する必要がある。 ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・Q1と同様 ・MTGや口頭でのやりとりまで踏まえて正確な数字を取得することが困難
M4		Q&Aの数（少なさ）	**26**	**10**	**6**	**0**	・母数の基準があいまいかもしれない 仕様書のページである場合の適宜が若干微妙 Excelだったらどうするのか、htmlを自動生成していたら？等疑問が残る。見積もり規模も同じ ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・案件内容に応じてQA数も変わるため、全てに適用できないと考えられる。 ・MTGや口頭でのやりとりまで踏まえて正確な数字を取得することが困難
M5		関連するQ&Aの数（多さ）	**11**	**24**	**6**	**0**	・RedmineなどのQAを適切に管理すれば実現可能。ただし、1 QA内で複数の話題をやり取りするなどがあると適切な数値が取れない ・QAは多岐に及び、メールや会話などを判断するのは現実で木でない ・MTGや口頭でのやりとりまで踏まえて正確な数字を取得することが困難 ・指標の定め方、取得方法のイメージがわかないです ・Q&Aを記載する粒度や人に依存してしまい、どのQ&A同士が関連するかが明確になり辛いと思われる
M6		仕様の確定が遅れた機能か（有無）	**13**	**23**	**5**	**1**	・議事録などから取得可能か？明確な判定基準がイメージできない。 ・個別要件単位の確認時間など開発側では認識するのは難しい。ただし開発側から問いかけた仕様については把握することは可能 ・遅れたかどうかではなく、十分に検討する時間があったかどうかの方が重要だと思う ・全仕様分の仕様確定日の取得はしてない ・遅れを測るために、何を持って遅れたのかを定義する必要があるため。
M7		開発組織内レビューまたは顧客レビューの回数・時間（少なさ、短さ）	**8**	**22**	**12**	**0**	・取得可能であるが、現在のプロジェクトでレビュー工数を適切に管理していない。 ・顧客側の数値をとることができるかが分らないため ・お客さまレビューについては、事前に工数・時間を把握しておくことを依頼しておかないと困難 ・時間計測が困難 ・顧客がどのぐらい時間をかけているかは正確に把握できない可能性が高い ・顧客がかけている時間は不明 ・顧客の工数について記録して頂く取組は実施したことがあるが、上手くいかったため。 ・レビュー工数・時間を記録でききれてない ・工数の付け方や収集方法を検討し、徹底させなければいけない。 ・DR工数は容易に実現できそうだが、多い少ないの基準が難しいと思われる ・時間計測は、人に依存しやすく、正確なデータが取得できづらいため。
M8		Q&Aの数に対する仕様変更回数（少なさ）	**11**	**27**	**5**	**0**	・M4と同じ
M9	プロジェクト全体	プロジェクト全体での仕様変更回数（多さ）	**26**	**14**	**2**	**0**	・M4と同じ

その他、仕様齟齬の予兆の検知に有効で、かつ、取得の実現性があるメトリクスがありましたら、記載してください。

・仕様変更時における他サブシステムなどへの影響調査に掛ける工数・時間（少なさ、短さ）
・① 新規機能数
・特にない（あれば活用している）
・Q1同様、テストシナリオの顧客レビュー指摘件数
・Q&Aのやり取り回数
※1つのQ&Aで何度もやり取りを繰り返しているケース
・派生元/兄弟/バリエーション違い等のプロジェクトとの仕様変更数の比較
・コード量
・CPUのROM/RAM使用量
・CPUの最大負荷率

その他（フリーコメント）

その他、ご意見等ありましたら、記載をお願いします。

・仕様齟齬の原因がある程度特定できているのでしょうか。

※ご協力ありがとうございました。

アンケート3.仕様齟齬予兆検知後の対処パターンと判断基準

SQIP第一分科会で討議している"仕様齟齬の発見手段パターンと最適な手段の選択基準"について、ご意見をお聞かせください。
（"3.仕様齟齬予兆検知後の「対処パターンと判断基準」"を参照してください。）
（選択式の項目は、解答欄のセルをクリックし、ドロップダウンリストから●を選択してください。）

◆各パターンの有効性について

Q 1 各対処パターンの"有効性"についてお答えください。

		非常に有効である	ある程度有効	あまり有効でない	全く有効でない	理由 （※任意："あまり有効でない" または、"全く有効でない"を選択された場合、その理由を記載してください。）
P1	顧客に詳細設計書を再度レビューしてもらう	**3**	**15**	**21**	**4**	・詳細設計書に改善がなければ、再度レビューしても同じ結果になると思われる。 ・顧客側への負担が大きく、ドキュメントのレビューでは齟齬の発見が効率的ではないため ・詳細設計は実装よりのことが多く、お客さまは内容を理解できない ・顧客が詳細設計書を理解するのが有効的でないと判断したため ・顧客はエンジニアではないので、詳細設計書を読ませてすべて確認してもらえるかは微妙 ・顧客にプログラムの詳細を伝えても意味がない。ユーザ（利用者）目線での説明・レビューが必要と思う。 ・お客様のリテラシーが低い・現業務繁忙等の理由により、詳細設計書の内容が理解できない、説明・理解してもらえる時間が無いケースが多い。 ・顧客の関与度・知識によってレビュー精度がばらつくので、品質が一定に測れない ・再レビューする際の観点を明確にしないと、あまり有効ではない ・顧客が詳細設計レベルをレビューして仕様齟齬が検知できるイメージがわかない(顧客次第) ・手戻りの大きな仕様確認に詳細設計書は不向き ・フォーマットによるが詳細設計書をベースに顧客が正しく内容を理解できる可能性が低いため ・詳細設計のレベルまで顧客はレビューできないと認識している ・詳細設計レベルの内容は、そもそも顧客が理解できないことが多い ・詳細設計書の粒度によるが、ロジックレベルの設計書だと、慣れない顧客がアレルギー反応を起こす可能性がある。 ・一度見せたものなので、観点をすり合わせる必要がある(見せ方を工夫する必要がある。そのままでは見せられない) ・詳細設計のレベルになると見れる方は限られる。 詳細設計までみれる顧客であれば有効 ・設計書では、顧客がイメージできず、合意しても再度仕様齟齬が発生する可能性がある。 ・詳細設計書はプログラマ向けである事が多いため顧客のレビューに向かない可能性あり ・詳細設計の粒度では顧客とは話しにくいかと思います ・そもそも、仕様齟齬が起こるケースは、詳細設計書のレビューができてない箇所で起こるのでは？ 再度レビューしたからといって、前回レビューできてない箇所をレビューできるとは限らない。レビューの仕方を工夫しないと、有効なレビューにならない。 ・詳細設計書レベルのものを顧客に見てもらって理解されるのか不明。 ・顧客が詳細設計書を理解できるとはかぎらない ・単純な分岐条件や、タイミング的な部分であれば意味があるかもしれないが、アプリケーションサーバ上にデプロイされたアプリ＋プラットフォーム２つを絡めた設計レビューは無理。
P2	テストケースやテストシナリオを作成して顧客に確認する	**17**	**23**	**3**	**0**	・顧客側への負担が大きく、ドキュメントのレビューでは齟齬の発見が効率的ではないため ・テストケースやシナリオを顧客に理解してもらうのが有効的でないと判断したため ・顧客は書いてあることを確認するが、あるべきとの比較でないと意味がない ・但し顧客の知識レベルに左右される ・UATやSTシナリオであれば有効 ・特にGUIではマスト
P3	気になる機能のみをスケジュール前倒しで実装（完成版）をおこない顧客にデモ実施	**23**	**14**	**6**	**0**	・開発側への負担が大きいため ・齟齬があった場合の手戻りが気になる ・一部の内容に特化しすぎる為。 ・完成版をつくるデメリットが大きすぎる ・完成版よりもP4のある程度のプロトタイプでデモするほうが確認が早いため早期につぶせる。 ・前倒しの度合いにもよるが、既に手遅れの場合も ・齟齬は単独機能というよりは実運用イメージで発覚する事が多いかと思います。単独機能で確認出来る内容であればモックアップで解決出来るかと思います。 ・意外と、気にしていない機能で認識齟齬が起こったりする場合も･･･ ・正常系のみ、テスト可。 排他系のテストや、多重アクセスなどの負荷テストは不可
P4	気になる機能のみをスケジュール前倒しでプロトタイプ（育てて最終的に使うもの）作成し顧客にデモ実施	**23**	**19**	**1**	**0**	・スケジュールが調整できれば可能かもしれない ・プロトタイプの完成度が高ければ有効
P5	気になる機能のみをスケジュール前倒しでモックアップ（最終的に捨てるもの）を作成し顧客にデモ実施	**15**	**24**	**4**	**0**	・モックアップと実態が異なるケースが多く、またお客様が操作してみて気づく要素が多々あるため。 ・捨てるものへの投資は無駄 ・P4を実施するならP5は不要 ・画面であれば有効。 ・特にGUIではマスト

その他、有効性のある手段があれば、記載してください。

・・再度、業務仕様レベルの資料を作成し顧客に確認してもらう
・①業務フローを担当者へ説明する
②ユーザーマニュアルの説明会をおこなう
・今の顧客では、P2を重点的に行っております。
・要件に対する機能仕様のウォークスルー
・P4の派生として、UI/UX部分に関しては、UT完了レベルで即顧客にみせると有効です。わざわざスケジュールを前倒しにする必要なし。結果、後工程の単純な指摘が激減する。
・ユーザ受入テストのシナリオなど、実際利用する顧客側が作成するテストパターンを提供していただき、それをSTに取り込むことで、ユーザの認識を受け入れ前に検知できる。
受入だけではなく、顧客がテストレビューを実施する前提であれば、テストファーストの考え方が有効だと考える。
・確認する機能の動作や動作結果を資料としてまとめ、ユーザ確認する。

◆各パターンの実現性について

Q 2 各対処パターンの"実現性"についてお答えください。

		容易に実現できる	実現できる	実現が困難	実現不可能	理由 （※任意："実現が困難" または、"実現不可能"を選択された場合、その理由を記載してください。）
P1	顧客に詳細設計書を再度レビューしてもらう	**5**	**12**	**24**	**2**	・顧客側に受け入れてもらえない可能性が高い ・詳細設計は実装よりのことが多く、お客さまは内容を理解できない ・顧客のリソースが割り当てられないケースが多い ・顧客に丸投げはできなそう ・細かい仕様にばかり目がいき、全体的な視点が抜けるケースが多い。 ・技術的な知見や開発アーキテクトの理解が必須となるため ・先方窓口が情報システム担当であれば実現可かと思いますが、ITに詳しくない事業担当者の場合は困難かと思います。 ・レビューの観点出しが必要 ・詳細設計レベルをレビューできる顧客はマレだと思う ・顧客側に時間的余裕がないケースが多い ・詳細設計レベルの記載と仕様を結びつけるのは困難 ・詳細設計書がない(作らない)場合が多い。 そもそも理解できないことが多い。 ・そもそも詳細設計書をレビューしない顧客がほとんど ・顧客の時間の確保が難しいため、確認したい要点を絞ったレビューであれば可能。 ・再レビューの必要性を顧客に理解してもらうことが前提 ・成果物をレビューできるスキルが顧客側に無いことが多い ・ユーザが詳細設計書を理解できるか疑問 ・詳細設計の粒度では顧客とは話しにくいかと思います ・詳細設計書にはシステムの内部設計が記述されており、顧客には理解が難しいことが多い ・詳細レベルをそのままレビューは工数的に厳しい。仕様をまとめた資料程度であれば可能。 ・顧客の協力を得られるか不明。 ・顧客が詳細設計書のDRに協力してくれるとは思えない ・顧客が詳細設計書を理解できるとはかぎらない ・あくまでも詳細設計書は、開発側の作業であり、ドキュメントであるため。
P2	テストケースやテストシナリオを作成して顧客に確認する	**9**	**29**	**4**	**1**	・顧客側に受け入れてもらえない可能性が高い ・業務観点のシナリオ・ケースであれば良いが、システム観点のものだと、設計成果物と同様に真意が伝わらない可能性がある ・成果物をレビューできるスキルが顧客側に無いことが多い ・社内技術やノウハウの流出に繋がるので、 ・顧客側もテストケースやテストシナリオが十分であることを判断しづらいため。 （何か起きた時に、確認しましたよね？というためのエビデンスにはなると思う）

P3	気になる機能のみをスケジュール前倒しで実装（完成版）をおこない顧客にデモ実施	**2**	**18**	**22**	**1**	・スケジュール変更を伴うため ・開発側のリソースが十分であり、最初から計画されていない限り難しい ・独立した機能でないと、機能間の連携があるため完成版の実装は困難 ・完成版にすることが現実困難と判断したため ・計画に組み込む必要があり、何を対象にすべきかの判断が必要 ・齟齬があった場合の手戻りが気になる ・独立機能であればよいが、他機能との関連性が強ければ短機能の実装では意味がなくなるため ・環境準備に時間を要する。フェージングしたのと近い状況になるため、工数が余計に要する。 ・ウォーターフォールモデルでの開発の場合、PJT全体との整合性が取りづらい ・プロジェクト期間に影響が出る為、難しい ・先行で完成版を作るリソースを割く反動が大きい為 ・仕様齟齬が発生するPJでは進捗も逼迫していることが多く、前倒し作業は困難なため。 ・気になる機能に紐づく他機能も同様に完成版が必要となると、早期に実施できず、あまり実現性を感じない。 ・コスト増加が許容されにくい。 プログラムの作り方が対応しにくいコーディングになっている ・気になる機能の事前作成用のスケジュールを予め計画として入れておく必要がある。その期間のコストを認めないお客様も多く実現不可能ではないが、コストが優位性の弊社では難しい部類にはいる。また、前依存する機能の考慮も並行して検討する必要があるため ・デモ実施のための工数が大きすぎる ・その機能だけでは動作しないことが多い。 １画面で完結し動作するような場合は実現可能。 ・デモ実施のスケジュールを最初に組み込めるかどうか ・納期が短いほど、実現が難しい ・完成後にデモだと発覚するタイミングは早まりますが手戻り時のロスも大きく防止にはあまり意味がないかと。 ・あらかじめ予定してたら、実現できるが。予算とそれに応じた体制があることが実現の前提。 ・完全スタンドアロンな機能でもない限り、部分的に機能を完成させるのは難しい。
P4	気になる機能のみをスケジュール前倒しでプロトタイプ（育てて最終的に使うもの）作成し顧客にデモ実施	**3**	**26**	**14**	**0**	・スケジュール変更を伴うため ・開発側のリソースが十分であり、最初から計画されていない限り難しい ・育てるフェーズにより、つぎはぎとなることが多く、困難 ・独立している機能に限る気がするが、これも手戻りがきになる ・結果的に捨てるパターンが多い。作る時期が早くやっつけの作りになることが多い ・ウォーターフォールモデルでの開発の場合、PJT全体との整合性が取りづらい ※アジャイルなら実現可能 ・仕様齟齬が発生するPJでは進捗も逼迫していることが多く、前倒し作業は困難なため。 ・コスト増加が許容されにくい。 プログラムの作り方が対応しにくいコーディングになっている ・気になる機能の事前作成用のスケジュールを予め計画として入れておく必要がある。その期間のコストを認めないお客様も多く実現不可能ではないが、コストが優位性の弊社では難しい部類にはいる。また、前依存する機能の考慮も並行して検討する必要があるため ・そこに注力した分、他の機能が遅れることがあることを顧客と合意することが前提 ・プロトタイプの場合、通常よりも費用がかかると考えられるため、最初からその想定で契約が必要。 ・納期が短いほど、実現が難しい ・あらかじめ予定してたら、実現できるが。予算とそれに応じた体制があることが実現の前提。 ・この時点での機能は、他機能との影響度合いも絡めてのデモとしなければならないので、機能のみとしては成立しないと思われる
P5	気になる機能のみをスケジュール前倒しでモックアップ（最終的に捨てるもの）を作成し顧客にデモ実施	**3**	**25**	**14**	**1**	・スケジュール変更を伴うため ・最終的に捨てる分を作る費用をお客さまに請求するのが困難 ・結果的に捨てるパターンが多い。作る時期が早くやっつけの作りになることが多い ・純粋に追加コストとなるため ・仕様齟齬が発生するPJでは進捗も逼迫していることが多く、前倒し作業は困難なため。 ・余分なコストになるため ・コスト増加が許容されにくい。 プログラムの作り方が対応しにくいコーディングになっている ・気になる機能の事前作成用のスケジュールを予め計画として入れておく必要がある。その期間のコストを認めないお客様も多く実現不可能ではないが、コストが優位性の弊社では難しい部類にはいる。また、前依存する機能の考慮も並行して検討する必要があるため ・プロジェクト計画段階で予定していれば問題ないが、仕様確認のためだけに作成し、捨ててしまうのでは、工数の無駄。 ・最終捨てるものについて、お金を出してください とい提案は自分が顧客なら受けない ・不具合アリの状態では難しいが、不具合解消してからでは遅い感じがする ・モックアップ作成の費用を確保出来ないと思う。 ・予算とスケジュールが顧客に了承してもらえるか

その他、実現性がある手段があれば、記載してください。

・①業務フローを担当者へ説明する
②ユーザーマニュアルの説明会をおこなう
詳細設計というよりは、要件仕様書みたいなものを作って確認する感じだと思う

判断基準について

Q３ 各対処パターンを選択するための"判断基準"として妥当性はありますか？（J1～J9）

		非常に妥当性である	ある程度妥当である	あまり妥当でない	全く妥当でない	理由 （※任意："あまり妥当でない" または、"全く妥当でない"を選択された場合、その理由を記載してください。）
J1	仕様齟齬を確認するためのモノを作る作業のコストの安さ	**16**	**23**	**3**	**1**	・作業コストの安さと連動することが理解できない。オフショアなど作業場所が異なると日々のコミュニケーションが取りづらく考えられる要因かを知れない。 ・この経費をどちらが持つかで変わる
J2	齟齬の発見のしやすさ＝「顧客に見せるモノ」の精度の高さ	**16**	**23**	**3**	**0**	・品質からイメージしにくい ・イコールにはならないと思う
J3	顧客が確認できる状態になるまでの期間の短さ	**17**	**26**	**0**	**0**	
J4	顧客側にかかる作業負荷の軽さ	**19**	**20**	**4**	**0**	・変更難度からイメージしにい ・負荷を恐れるか、齟齬を解消できないのア実状 ・仕様齟齬は基本顧客の責任と考える
J5	開発側に必要な技術・ノウハウの難度の低さ	**12**	**26**	**5**	**0**	・変更難度からイメージしにくい ・開発スキルよりも仕様の難易度が重要 ・あまり必要だとは感じなかった。なぜ必要なのか分からない。 ・仕様齟齬を顧客に判断してもらうことの重要性を考慮すると
J6	ウォーターフォール型開発からの移行のし易さ	**4**	**25**	**13**	**1**	・判断基準の意味が理解できていません ・すみません。判定できません。項目の意味が分りませんでした。 ・開発手法はまり関係ないと思われる ・ウォーターフォール以外の手法を使っている場合は考慮する必要が無いので、汎用的な判断基準としては成り立たない ・判断基準がよくわからない ・開発手法を移行することは容易ではないので、全体の流れは変えず一部の機能だけ例えばプロトタイプ型を適用するイメージ ・あまり必要だとは感じなかった。なぜ必要なのか分からない。 ・請負からアジャイル等に気軽に変えると痛い目にあう ・できれば開発方式に依存しない方がいいので ・品質保証体制は崩せない ・開発モデルは、ものによって向き不向きがあり、良し悪しを把握する基準になりづらいため、傾向は見えずらいと思う。
J7	他の工程、全体スケジュールに与える影響度の低さ	**24**	**17**	**1**	**1**	・すみません。判定できません。項目の意味が分りませんでした。 ・仕様齟齬は基本顧客の責任と考える。全体スケジュールへの影響も顧客が考慮すること
J8	顧客との調整（協力の引き出し易さ、顧客の時間の融通の）のし易さ	**22**	**19**	**2**	**0**	・変更難度からイメージしにくい ・仕様齟齬は基本顧客の責任と考える
J9	仕様齟齬が見つかった場合の無駄の少なさ	**16**	**24**	**3**	**0**	・発生する仕様変更に依存する ・仕様齟齬は基本顧客の責任と考える

その他、妥当性のある判断基準があれば、記載してください。

◆対処パターンと判断基準の優先順位の妥当性について

Q４ 前シート「３．仕様齟齬予兆検知後の対処パターンと判断基準」の優位順付け（A～D）について、妥当性はありますか？（J1～J９）

		非常に妥当性である	ある程度妥当である	あまり妥当でない	全く妥当でない	理由 （※任意：”あまり妥当でない” または、”全く妥当でない”を選択された場合、その理由を記載してください。）
J1	仕様齟齬を確認するためのモノを作る作業のコストの安さ	**7**	**27**	**7**	**2**	・実際に実装するものが一番コストがかかると思われます。 ・Dは実装（完成版）である ・完成版を早めに動かす場合、環境準備等を前倒しで実施する必要がある。サービス利用等の場合、早くからランニングがかかる。また、フェージングするのに近い考えになるので作業コストも必要になる。 ・モックの精度によってコストは変動する ※紙芝居レベルならテストケース見せるよりコストが安い可能性もある ・スケジュールを前倒すのであれば、コスト増にはならない、ただ後々捨ててしまうものについては純増となる ・③がBで④がDが逆な気がする ・モックアップの方が簡単？ ・テストケース、テストシナリオの作成確認Aとなっていますが、顧客側にもそこから、仕様ミスの判断は難し、開発側のエビデンス取りにしかならないと考えます。(以降同様)
J2	齟齬の発見のしやすさ＝「顧客に見せるモノ」の精度の高さ	**13**	**20**	**8**	**1**	・テストケースやテストシナリオでは齟齬の発見は難しい ・②のA：テストケースやテストシナリオを確認できる顧客は多くないと思われ、「顧客に見せるモノ」なのか疑問だから。 ・テストケース・テストシナリオを見せる場合と実物を見せるもので同一精度にはならない（明らかに実物を見せた方が精度が高い） ・②はBかCくらい。 ・テストケースやシナリオよりプロトタイプやモックアップのほうが有効と考える。 ・テストケースでは齟齬は発見しにくいのでは？ ・「気になる機能のみをスケジュール前倒しで実装（完成版）をおこない顧客にデモ実施」これがAではないと思う
J3	顧客が確認できる状態になるまでの期間の短さ	**16**	**23**	**4**	**0**	・プロトタイプのほうが、モックアップより優位性が高いと感じる ・モックアップの方が簡単？ ・各対処が実施され顧客が確認できるまでの作業がイメージできませんでした。
J4	顧客側にかかる作業負荷の軽さ	**9**	**24**	**10**	**0**	・実装とモックアップは同じではないはず （データは異なるのでは？） ・「気になる機能のみをスケジュール前倒しでプロトタイプ･･･」C判定なのが妥当でないと判断したため ・プロトタイプがなぜDか？ ・プロトタイプ（A)とモック（C)でランクが分かれているが、どちらも一定の制限は加わるので作業負荷そのものにはそこまでの差異は無いように思う ・テストケース・シナリオのレビューはB or Cなのでは ・②はBくらい。 ・詳細設計レビューは作業負荷があると考える。 プロトタイプはあまり負荷はないと考える。(モックアップと同じ) ・実装（完成版）～モックアップは作る側の負担の重さの違いでユーザー側からはそれほど差が無いように思います。 ・「気になる機能のみをスケジュール前倒しで実装（完成版）をおこない顧客にデモ実施」でその他の機能が造られていない状態で完成品と同等の動作させることができるのか？
J5	開発側に必要な技術・ノウハウの難度の低さ	**9**	**22**	**11**	**1**	・新技術を利用していたり、複雑な機能の場合、実装をする場合は難易度が高いではないでしょうか？ ・実装は技術が必要なはず ・「プロトタイプを作成して･･･」のみD判定としたこと、「完成版を作成して･･･」(A)との差が妥当でないと判断したため ・プロトタイプの場合も将来的に実装するものなので、実装版に求められる技術・ノウハウと同等と考えられる。 ・④はDではないと思う ・⑤はBかCくらい。 ・プロトタイプのAの認識 ・どれも同じ？ ・プロトタイプ作成の難易度は低い ・各対処が実施され顧客が確認できるまでの作業がイメージできませんでした。 ・「気になる機能のみをスケジュール前倒しで実装（完成版）をおこない顧客にデモ実施」完成版を作るには品質保証まで考慮して開発する必要がある。作成負荷が高いのでは?
J6	ウォーターフォール型開発からの移行のし易さ	**5**	**23**	**14**	**0**	・すみません。判定できません。 項目の意味が分りませんでした。 ・④のD：今日において、プロトタイプとウォータフォールは相反するものではないと考えているから。 ・「プロトタイプを作成して･･･」のみD判定としたこと、「完成版を作成して･･･」(A)との差が妥当でないと判断したため ・先行開発方式への切替は、現場の抵抗が一定発生する。 ・判断基準がよくわからない ・プロトタイプのAの認識 ・請負からアジャイル等に気軽に変えると痛い目にあう ・どれも同じ？ ・各対処が実施され顧客が確認できるまでの作業がイメージできませんでした。 ・「気になる機能のみをスケジュール前倒しで実装（完成版）をおこない顧客にデモ実施」でその他の機能が造られていない状態で完成品と同等の動作させる仕組みを作るので、製品開発プロセスから逸脱する。
J7	他の工程、全体スケジュールに与える影響度の低さ	**11**	**23**	**8**	**0**	・前倒し実装はスケジュールへの影響が大きいと思います。 ・すみません。判定できません。 項目の意味が分りませんでした。 ・モックはドキュメント・製品とは別につくるので、作業量が純増するものと思っており、スケジュールに与える影響が一番高いと思う ・J1と同じ優先順位なのではないか ・プロトタイプDとモックアップBの評価が反対の認識 ・どれも同じ？ ・各対処が実施され顧客が確認できるまでの作業がイメージできませんでした。 ・モックアップを作る事は当初の開発スケジュールに無いため、”B”では無いと思う
J8	顧客との調整（協力の引き出し易さ、顧客の時間の融通の）のし易さ	**9**	**26**	**6**	**2**	・テストケースやテストシナリオを細かく見てもらうのは相当時間が必要なはず ・②のA：テストケースやテストシナリオを確認できる顧客は多くないと思われ、「協力の引き出し易さ」に疑問をもつから。 ・実装版・プロトタイプ・モックなど「モノ」を見せる事とテストドキュメントを見せる事に対して顧客との調整は同等にはならない。 また、テストドキュメントと設計書については調整難易度としては変わらない（どちらもモノを見せる場合と比べて高い）と思われる ・J4と同様 ・プロトタイプがBではなくAの認識 ・モックアップの作成を契約時に盛り込んでいれば妥当と思うが、そうでなければ、”B”と考える。 ・どれも同じ？ ・「テストケースやテストシナリオを作成して顧客に確認する」で顧客は仕様齟齬を確認し易いのかは疑問
J9	仕様齟齬が見つかった場合の無駄の少なさ	**8**	**31**	**4**	**0**	・④はBくらい。 ・テストケースやシナリオは作成するタイミングにより、無駄の量が変わる。 ・テストケース作成工程は製造工程より後である事が多いため無駄の少なさはプロトタイプの方が少ないかも

その他、ご意見があれば、記載してください。

・詳細設計書に優位順がないのはなぜか
・実装アーキテクチャを無視すればプロトタイプが一番無駄の少なさと作成負担のバランスがいいと思います。今は生産性の悪い設計思想・アーキテクチャなのでアレですが･･･。
・対処パターンの実現性がありき。

本研究の全体を通して

Q５ 仕様齟齬に関する問題は開発の上流工程で対応することが一般的ですが、本研究会のように開発の途中段階で検知、対応する手法について研究していく意義はあると思いますか？

ある	ある程度ある	あまりない	ない	理由
24	**19**	**0**	**0**	

その他（フリーコメント）

その他、ご意見等ありましたら、記載をお願いします。

・対処パターンを検討するに当たっては、対処方法に対する顧客側の納得感も重要なのではと考えています。
・仕様齟齬が早く検出され、お客さまも開発者もハッピーになれることに貢献する研究成果がでることを期待しています。
・開発のプロに仕事依頼が来ている訳ですから仕様部分については仮に説明が無くても、こちらから判断を委ねる材料を提供し明確化することが重要であると思う。また、問題が発生した場合は状況を理解しある関係が重要。そのためにも全体のリソースと優先度の意味を顧客に理解いただくことが大事と認識している。
・開発途中段階での検知対象の中には、全フェーズの終了クライテリアに組み込んだ方が良いものも出てくると思うので、そちらにも生かせるとよいかと思います。
・仕様齟齬が発生するケースの中に顧客と開発者の観点のほかに顧客内での仕様認識に対する齟齬があり、それらが最終的な顧客の受け入れ時に発覚するケースが少なくないと感じます。その視点が今回のメトリクスでは検知しづらくなっているような気がしました。
・請負であっても、工程が進まないと判断できないもの、実物があった方が理解が深まるものについては、半完成時点で顧客にみせるべき。仕様齟齬を前向きにとらえ、工期中に取り込める”未来の仕様変更”だと考えたほうが、リリース後の利用価値、顧客評価が高まる。
・プロジェクトを運営していく中で、キナ臭さを感じる「予感」的なものが、経験のあるPMであれば皆あるのではないでしょうか？その点が、メトリクス化されれば、経験の浅いPMにも武器になると思います。SQIP第一分科会での研究成果を楽しみにしています。
・どのフェーズでも仕様齟齬の予兆検知ができるのであれば非常に役に立つと思います。ただ、後工程になればなるほど調整等が難しくなるため、できるだけ早い段階で検知できる方法を研究いただければと思います。
・期待してます。
・顧客へのデモにつきましては、完成版・プロト・モックについて、「機能のみ」という言葉が当てはめ辛い場合(調停が必要で機能のみでのは成り立たない物等)があると思われますので、定義が難しいかなと感じました。
・情報セキュリティが製品に求められてくると、顧客も要求仕様（セキュリティ仕様）をFIXできない可能性が在る。その場合、顧客ー開発チームで仕様のFIXに向け協力していくプロセスが必要と考える。

※ご協力ありがとうございました。