付録1_認知バイアスの自己認識アンケート結果(個別)

頻度

1	全くない
2	まれにある
3	ときどきある
4	よくある
5	いつもそうだ （非常に頻繁にある）

役割

SM	スクラムマスター
PO	プロダクトオーナー
DevL	開発リーダー
Dev	開発担当
QA	品質保証担当

No.	カテゴリ1	カテゴリ2	具体例	役割	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
					SM	SM	SM	SM	SM	DevL	DevL	PO	Dev	DevL	Dev	Dev	Dev	Dev	QA	QA	Dev	QA	QA	QA	QA	Dev	Dev	QA	PO	PO	PO	PO	DevL	DevL	DevL	Dev	Dev	Dev	DevL	DevL	DevL	Dev
1	経験とヒューリスティクス（経験則と思い込み）	正常性バイアス	・「今までこの手の修正で問題が起きたことはないから，今回も大丈夫だろう」と，潜在的なリスクを無視してしまう．		3	3	2	2	2	2	4	4	3	3	3	2	3	3	3	3	3	2	2	2	2	2	1	4	3	4	4	3	2	2	4	4	2	2	2	3	3	1	2
2			・軽微な修正だと判断し，「このくらいの変更なら，詳細な影響調査は不要だろう」と自己判断でプロセスを省略してしまう．		1	1	2	1	1	2	2	4	3	3	2	2	3	4	2	2	1	1	3	1	1	1	2	3	4	1	2	3	2	2	2	2	3	2	2	3	2	2	3
3		利用可能性ヒューリスティック	・過去に類似の修正で障害が発生した経験がないため，「このパターンの修正は安全だ」と思い込んでしまう．		3	2	2	1	3	2	5	5	4	2	3	4	2	3	2	4	3	2	2	2	1	2	2	3	2	3	4	2	2	4	2	4	2	2	2	3	1	1	2
4			・直近で問題なくリリースできた成功体験に引きずられ，「今回も同じ手順でやれば問題ない」と安易に考えてしまう．		2	3	2	3	3	1	4	5	4	3	2	3	2	4	3	4	3	3	3	3	3	3	2	1	4	3	4	3	4	4	2	4	4	2	2	2	3	2	1	3
5		アンカリング	・最初に見積もった影響範囲（例えば「影響があるのはこの2画面だけ」）に固執し，後から発覚した他の影響箇所を軽視してしまう．		1	1	1	1	1	1	1	2	2	2	1	2	2	4	2	2	1	2	4	2	2	1	3	1	2	1	3	1	2	2	2	4	2	3	1	2	1	2	2
6		※有識者の声を信じる	・特定の技術的有識者の「この修正の影響は軽微だ」という最初の発言を鵜呑みにし，他の可能性を十分に検討しない．		1	3	3	4	2	3	3	3	2	3	3	4	3	5	4	3	1	4	4	3	3	3	3	1	3	2	4	2	2	4	3	3	3	3	2	2	2	2	4
7		確証バイアス	・「このモジュールは独立性が高いはずだ」という思い込みに基づき，その仮説を支持する情報ばかりを探し，反証する情報（実は依存関係があるなど）を無視する．		3	1	3	2	1	1	1	2	1	3	2	1	2	3	2	2	1	1	3	2	2	1	2	1	4	1	3	2	3	3	1	2	2	2	2	3	2	1	2
8			・自分が修正した箇所については，「問題ないはずだ」という先入観でレビューを行い，問題点を見逃してしまう．		1	2	3	1	3	1	3	1	1	2	1	1	1	2	3	2	1	2	1	2	1	2	1	2	3	1	4	1	2	2	4	1	2	2	2	3	1	2	3	2	2
9		代表制ヒューリスティック	・「この修正はUIの文言変更だけだから，サーバー側の処理に影響があるはずがない」と，典型的なイメージで判断し，実際にはその文言がログ出力や特定処理のキーになっている可能性を調査しない．			2	2	2	3	2	4	3	1	3	2	3	2	5												2	2	4	4	3	3	3	4	2	3	3	1	2	1	2
10			・「このバグは，前に経験したあの単純なバグと症状が似ている．原因も同じだろう」と決めつけ，より深刻な根本原因を見逃してしまう．		1	1	2	1	2	2	1	3	4	4	2	2	2	4	2	3	1	2	3	1	3	1	2	2	3	2	3	2	3	2	2	3	2	1	2	2	1	1	2
11	自信と自己評価	ダニング＝クルーガー効果	・経験の浅いエンジニアが自身のスキルを過信し，「このくらいの実装なら，影響範囲は完全に把握できている」と判断してしまう．		1	1	4	1	1	1	1	1	3	2	1	1	1	2	1	2	1	1	3	1	1	1	3	3	1	2	2	2	2	2	2	2	2	2	2	1	2	1	1	2
12			・新しい技術要素について少し触っただけで全体を理解した気になり，複雑な依存関係を見落としてしまう．		1	1	3	1	1	1	2	1	1	3	1	2	1	5	3	2	1	1	2	1	1	1	4	1	1	1	1	1	2	2	1	2	2	2	1	2	3	1	2
13		楽観主義バイアス	・「テストで問題が出なければ，本番でも問題は起きないだろう」と，予期せぬ本番環境固有の問題が発生する可能性を過小評価する．		1	2	3	1	1	1	1	4	1	4	2	1	3	2	4	2	1	2	3	2	2	1	3	3	2	2	4	1	3	2	2	2	4	2	1	3	4	1	3
14			・「何か問題があっても，すぐに修正（リカバリ）できるだろう」と，障害発生時の影響の大きさを楽観的に考えてしまう．		1	1	4	1	2	2	1	2	1	3	2	2	2	5	1	3	1	1	1	1	1	2	3	3	2	1	3	1	1	1	2	2	2	1	1	2	2	1	2
15	社会的要因（集団心理と同調圧力）	同調圧力・バンドワゴン効果	・レビュー会議で他のメンバーが「問題ない」と言っているため，疑問に思う点があっても「自分が間違っているのかもしれない」と考え，指摘せずに流してしまう．		1	1	3	1	1	3	1	4	3	4	3	4	4	1	1	4	2	3	2	4	1	2	1	1	4	1	3	1	2	1	4	2	2	5	1	4	1	2	2
16			・チームリーダーや経験豊富なエンジニアの意見に，異論を唱えにくい雰囲気があり，潜在的なリスクが見過ごされる．		1	1	3	1	1	4	1	4	4	3	4	4	3	1	3	4	1	2	1	1	1	1	1	1	3	1	2	2	3	2	4	2	1	5	1	3	1	3	2
17		権威バイアス	・技術顧問やアーキテクトが「その修正による影響はない」と言ったため，他のメンバーが疑問を感じつつも，それ以上の影響範囲の調査を行わなかった．		2	1	3	3	4	2	1	5	2	3	2	2	3	3	1	3	1	2	2	4	1	1	3	1	3	2	2	2	3	1	4	2	1	5	1	3	1	2	2
18			・経験豊富なベテランエンジニアが「自分の経験上，この変更で問題が起きたことはない」と発言したため，チーム全体がその意見を鵜呑みにしてしまい，潜在的なリスクが見逃された．		1	2	3	2	2	2	1	5	2	3	2	2	2	3	1	2	1	1	1	3	2	1	2	1	3	3	3	2	3	1	3	2	1	5	2	3	1	2	2
19		集団浅慮（グループシンク）	・リリースが遅延している状況で，「これ以上問題点を指摘して，全体の和を乱したくない」という心理が働き，チーム全体としてリスクの高い判断を下してしまう．		1	1	3	2	1	2	1	4	2	2	2	1	2	4	3	3	2	1	3	2	1	1	2	1	4	1	3	2	2	1	4	3	1	5	1	3	1	1	2
20			・チーム内で「我々のチームは優秀だから大丈夫だ」という過信が生まれ，批判的な視点でのレビューが疎かになる．		1	1	2	1	1	1	2	1	1	2	1	1	1	2	1	2	1	1	2	1	1	1	1	1	2	1	1	1	2	1	1	2	1	2	1	2	2	1	2
21		内集団バイアス	・「我々のチームのコードは品質が高いが，他チームが作ったモジュールは品質が低い可能性がある」と考え，自チームが担当する範囲の影響調査は甘くなり，他チームとの連携部分のリスクを見落とす．		1	1	2	1	1	1	3	1	1	2	1	1	1	2	2	1	1	1	2	1	1	1	1	1	2	1	1	1	2	2	1	2	1	2	1	1	2	1	2
22			・「以前，あのチームのレビューは形式的だったから，今回も深く考えなくていいだろう」と，特定のグループに対する先入観で対応を変えてしまう．		1	1	3	2	2	1	4	1	1	2	1	1	1	2	3	1	1	1	2	1	1	1	1	1	3	2	2	1	3	1	2	2	1	2	1	1	1	1	2
23	現状維持と損失回避	現状維持バイアス	・「これまでこのやり方で問題なかったのだから，新しい影響範囲の特定手法を取り入れる必要はない」と，プロセスの改善を避けてしまう．		1	2	2	1	1	3	3	1	2	2	3	2	2	3	4	4	3	2	1	2	2	3	4	3	3	2	3	4	1	1	4	2	2	4	2	3	1	2	3
24			・影響範囲が広いと判明すると，修正作業が大規模になり大変だと感じ，「できれば影響範囲は小さいままであってほしい」と考え，調査を途中で打ち切ってしまう．		1	1	2	1	1	1	2	1	1	2	1	2	1	2	2	2	1	1	2	2	1	1	2	1	2	1	3	1	1	1	1	2	1	4	1	2	1	1	3
25		損失回避性	・影響範囲を広めに特定することで，追加のテスト工数やスケジュール遅延が発生することを恐れ，意図的に影響範囲を狭く見積もってしまう．		1	1	3	1	1	1	1	1	2	2	1	1	1	2	3	3	1	2	1	1	1	1	2	1	3	1	2	2	2	1	1	2	1	4	1	4	1	1	2
26			・「もしここで新たな問題点を指摘したら，リリースが延期になって自分の評価が下がるかもしれない」と考え，リスクの報告を躊躇する．		1	1	4	1	1	1	1	1	2	2	2	1	1	1	3	2	1	1	1	1	1	1	1	1	2	1	2	3	2	1	2	2	1	4	1	4	1	1	2
27		サンクコスト効果	・「この機能の開発には既に多くの時間を費やしてきた．今さら影響範囲が広いとは言えない」と考え，調査を簡略化してしまう．		1	1	3	1	1	1	4	1	2	2	2	2	2	1	1	2	1	2	1	1	1	1	1	1	3	1	4	1	2	1	1	3	2	4	1	2	1	1	2
28			・特定の技術的アプローチが問題だと分かりつつも，「ここまで作り込んでしまったのだから，後戻りはできない」と判断し，根本的な修正を避けてしまう．		3	2	3	3	4	3	2	1	2	4	2	3	2	2	1	3	3	2	2	1	1	1	3	1	3	2	3	3	1	3	2	3	3	4	2	1	1	2	2

付録2_認知バイアスの自己認識アンケート結果(コメント)

No.	カテゴリ1	カテゴリ2	具体例	コメント
1	経験とヒューリスティクス（経験則と思い込み）	正常性バイアス	・「今までこの手の修正で問題が起きたことはないから，今回も大丈夫だろう」と，潜在的なリスクを無視してしまう．	・シーケンスのとある瞬間にメモリが故障すると、復帰できなくなる不具合を経験したことがある。シーケンス検討がこれで十分と思い込み、故障パターン(潜在的なリスク)の考慮が不足していた。 ・軽微で少量の機能追加や変更を実施した。その追加/変更内容自体は過去にも問題が起きたことはないため大丈夫と思いこんだが、実はその追加/変更がきっかけで処理負荷オーバーやメモリ・スタック等のオーバーフローを引き起こすトリガーとなってしまった。 ・似た処理が2箇所にあったため、消しても問題ないと思い、一方を削除した。しかし、実は一部異なる処理が入っていたため、必要な処理も消されてしまい、問題を混入してしまった。
2			・軽微な修正だと判断し，「このくらいの変更なら，詳細な影響調査は不要だろう」と自己判断でプロセスを省略してしまう．	・プロセスで規定されてるならば、省略はしない。省略したいと思うことは良くあるが、自己判断ではやらない。 ・リーダがしっかりしてたら、省略せずに打ち上げる。リーダも責任は自分にあるからちゃんと言ってね！を言ってくれる。そういう風土なら、省略せずに言いそう。リーダがしっかりしてなくて、頼りないなら、担当者が自分で判断して、省略してしまうこともありそう。 ・画面設計で軽微なUI修正を実施した。事前に顧客への確認・合意も取って変更したが、別の顧客からクレームを受けた。 (一部の顧客から了承を得られたので、他の顧客への調査は不要だろうと判断した。多くの顧客への確認が不足していた) ・テーブル設計でデータを1つ追加した。軽微な修正だと判断し、詳細な影響調査をしていなかった結果、IDなどの番号が1つズレた影響で不具合に繋がった。 ・通信メッセージのID番号やデータ内容を変更した。軽微な修正で主要な通信相手との整合は取れているから大丈夫と思い、影響調査を詳細に行わなかった結果、旧メッセージIDやデータを参照している通信相手がおり、通信途絶や異常誤検出に繋がった。
3		利用可能性ヒューリスティック	・過去に類似の修正で障害が発生した経験がないため，「このパターンの修正は安全だ」と思い込んでしまう．	・横展開するときに、安全だと思い込んでしまいそうだとも思う。真面目に考えると、横展開する先の周辺で影響の有無を見ると思う。
4			・直近で問題なくリリースできた成功体験に引きずられ，「今回も同じ手順でやれば問題ない」と安易に考えてしまう．	・enumが2つあり、暗黙的なつながりがあり、両方修正しなければならないのだが、知らずに片方だけ修正してしまった。新しいモデルは(たまたま)問題なかったため、古いモデルも問題ないと判断してしまった。これに関しては、新しいモデルは有識者レビューをしており、有識者は両方のテーブルを修正しないといけないことを認識していた。しかし、古いモデルは有識者レビューを省略してしまった。
5		アンカリング	・最初に見積もった影響範囲（例えば「影響があるのはこの2画面だけ」）に固執し，後から発覚した他の影響箇所を軽視してしまう．	-
6		※有識者の声を信じる	・特定の技術的有識者の「この修正の影響は軽微だ」という最初の発言を鵜呑みにし，他の可能性を十分に検討しない．	・誰かが言ったから自分の責任にならないと思ってしまう。その時の状況にもよると思う。余裕があれば調べよう思うが、余裕がなければ調べることを省略するかもしれない。 ・スクラムでプロダクトオーナーの立場だと詳細を把握している開発者が発言したことを信じてしまう可能性が高い。テストは実施するので、そこでNG結果が出て気づくという手戻りに繋がるリスクはある。 ・自身の知見・経験が浅い、専門分野違いの場合、有識者の発言に対しての理解が追い付かず、その発言を鵜呑みにしたり、他の可能性の検討が漏れたりする。
7		確証バイアス	・「このモジュールは独立性が高いはずだ」という思い込みに基づき，その仮説を支持する情報ばかりを探し，反証する情報（実は依存関係があるなど）を無視する．	-
8			・自分が修正した箇所については，「問題ないはずだ」という先入観でレビューを行い，問題点を見逃してしまう．	-
9		代表制ヒューリスティック	・「この修正はUIの文言変更だけだから，サーバー側の処理に影響があるはずがない」と，典型的なイメージで判断し，実際にはその文言がログ出力や特定処理のキーになっている可能性を調査しない．	-
10			・「このバグは，前に経験したあの単純なバグと症状が似ている．原因も同じだろう」と決めつけ，より深刻な根本原因を見逃してしまう．	-
11	自信と自己評価	ダニング＝クルーガー効果	・経験の浅いエンジニアが自身のスキルを過信し，「このくらいの実装なら，影響範囲は完全に把握できている」と判断してしまう．	-
12			・新しい技術要素について少し触っただけで全体を理解した気になり，複雑な依存関係を見落としてしまう．	-
13		楽観主義バイアス	・「テストで問題が出なければ，本番でも問題は起きないだろう」と，予期せぬ本番環境固有の問題が発生する可能性を過小評価する．	・テスト環境と本番環境の正確な違いを把握できておらず、同様に上手くいくと思っていたが、本番環境では設定を変更しないといけなかったり、環境依存(ビルドバージョン)の差を見落としたりして不具合につながった。 ・受け入れでのバグ。HTMLのタグがついてなかった。顧客の端末でログインのシングルサインオンがそのタグを認識できないという不具合となった。 ・受け取ったHTMLを一つのブラウザでしか動作確認していなかったため他のブラウザで不具合となった。 ※HTML（HyperText Markup Language：ハイパーテキストを記述するためのマークアップ言語）
14			・「何か問題があっても，すぐに修正（リカバリ）できるだろう」と，障害発生時の影響の大きさを楽観的に考えてしまう．	・変更規模が大きければ、リカバリが大変だろうと直感的に思い、楽観的には捉えないだろう。変更規模が小さいと、変更規模が影響も少ないだろうと思って甘く見てしまいそう。 ・アジャイル開発だとフィードバックをもらってすぐに修正すれば良いだろうというマインドになりやすいかもしれない。 (問題が出ても謝罪してすぐ適応すればなんとかなるというマインド)
15	社会的要因 （集団心理と同調圧力）	同調圧力・バンドワゴン効果	・レビュー会議で他のメンバーが「問題ない」と言っているため，疑問に思う点があっても「自分が間違っているのかもしれない」と考え，指摘せずに流してしまう．	-
16			・チームリーダーや経験豊富なエンジニアの意見に，異論を唱えにくい雰囲気があり，潜在的なリスクが見過ごされる．	-
17		権威バイアス	・技術顧問やアーキテクトが「その修正による影響はない」と言ったため，他のメンバーが疑問を感じつつも，それ以上の影響範囲の調査を行わなかった．	-
18			・経験豊富なベテランエンジニアが「自分の経験上，この変更で問題が起きたことはない」と発言したため，チーム全体がその意見を鵜呑みにしてしまい，潜在的なリスクが見逃された．	-
19		集団浅慮（グループシンク）	・リリースが遅延している状況で，「これ以上問題点を指摘して，全体の和を乱したくない」という心理が働き，チーム全体としてリスクの高い判断を下してしまう．	・集団浅慮はあると思う。何か問題を起こしたくないと直感的に思う。明らかに問題が明白なものは発言するが、問題かどうかが微妙なラインの場合は、言わないこともあるかもしれない。 ・開発者の立場で考えると、どんな小さい問題であっても言うと思う。立場が関係していると思う。 ・プロジェクトマネジメントしている立場だと、何かしら理由をつけて問題を無い方向に誘導できないか、考えてしまいそう。
20			・チーム内で「我々のチームは優秀だから大丈夫だ」という過信が生まれ，批判的な視点でのレビューが疎かになる．	-
21		内集団バイアス	・「我々のチームのコードは品質が高いが，他チームが作ったモジュールは品質が低い可能性がある」と考え，自チームが担当する範囲の影響調査は甘くなり，他チームとの連携部分のリスクを見落とす．	-
22			・「以前，あのチームのレビューは形式的だったから，今回も深く考えなくていいだろう」と，特定のグループに対する先入観で対応を変えてしまう．	-
23	現状維持と損失回避	現状維持バイアス	・「これまでこのやり方で問題なかったのだから，新しい影響範囲の特定手法を取り入れる必要はない」と，プロセスの改善を避けてしまう．	・既にDRBFMやFMEA等の影響範囲特定手法を使用しており、それらの対応だけでも相応の時間も掛かるし、過去の実績もある。なので、これ以上新しい手法を取り入れる必要はない、取り入れたくないという考えてに至ってしまう。 ・モデル展開終盤で機械的に作業してしまった ※DRBFM（Design Review Based on Failure Mode：故障モードに基づく設計レビュー） ※FMEA（Failure Mode and Effects Analysis：故障モード影響解析）
24			・影響範囲が広いと判明すると，修正作業が大規模になり大変だと感じ，「できれば影響範囲は小さいままであってほしい」と考え，調査を途中で打ち切ってしまう．	・調査しなくても頭の中で問題なさそうと思ったら、調査を実行しないことはある。その判断を謝ったりすると、問題が発生して大変なことになることもあるかもしれない。
25		損失回避性	・影響範囲を広めに特定することで，追加のテスト工数やスケジュール遅延が発生することを恐れ，意図的に影響範囲を狭く見積もってしまう．	-
26			・「もしここで新たな問題点を指摘したら，リリースが延期になって自分の評価が下がるかもしれない」と考え，リスクの報告を躊躇する．	・開発者としては、レビューに参加している全員にも関係するのだから、自分だけの責任じゃないと思って、躊躇することはなさそう。ただ、プロジェクトマネジメントする立場の人は、頭の中で何か問題とならないストーリが立てられるか、その道があるかを一生懸命考えてしまう。インプットである要求仕様に落ち度がないかを考えてしまう。
27		サンクコスト効果	・「この機能の開発には既に多くの時間を費やしてきた．今さら影響範囲が広いとは言えない」と考え，調査を簡略化してしまう．	・モデル展開終盤で機械的に作業してしまった
28			・特定の技術的アプローチが問題だと分かりつつも，「ここまで作り込んでしまったのだから，後戻りはできない」と判断し，根本的な修正を避けてしまう．	・実際に評価し結果を確認した時に、結果に対する指摘があった場合、「これやりなおすのか？そんな時間はもう残ってないよ。」と考え、次フェーズに先送りすることがある。

付録3_認知バイアスの自己認識アンケート結果(まとめ)

頻度
1 全くない
2 まれにある
3 ときどきある
4 よくある
5 いつもそうだ （非常に頻繁にある）

回答数 39

No.	カテゴリ1	カテゴリ2	具体例	回答結果 1	2	3	4	5	平均値	標準偏差
1	経験とヒューリスティクス（経験則と思い込み）	正常性バイアス	・「今までこの手の修正で問題が起きたことはないから，今回も大丈夫だろう」と，潜在的なリスクを無視してしまう.	2	16	14	7	0	2.397	0.896
2			・軽微な修正だと判断し，「このくらいの変更なら，詳細な影響調査は不要だろう」と自己判断でプロセスを省略してしまう.	10	17	9	3	0		
3		利用可能性ヒューリスティック	・過去に類似の修正で障害が発生した経験がないため，「このパターンの修正は安全だ」と思い込んでしまう.	4	19	8	6	2	2.731	1.009
4			・直近で問題なくリリースできた成功体験に引きずられ，「今回も同じ手順でやれば問題ない」と安易に考えてしまう.	3	10	15	10	1		
5		アンカリング ※有識者の声を信じる	・最初に見積もった影響範囲（例えば「影響があるのはこの2画面だけ」）に固執し，後から発覚した他の影響箇所を軽視してしまう.	15	18	3	3	0	2.359	1.025
6			・特定の技術的有識者の「この修正の影響は軽微だ」という最初の発言を鵜呑みにし，他の可能性を十分に検討しない.	3	9	18	8	1		
7		確証バイアス	・「このモジュールは独立性が高いはずだ」という思い込みに基づき，その仮説を支持する情報ばかりを探し，反証する情報（実は依存関係があるなど）を無視する.	13	16	9	1	0	1.923	0.844
8			・自分が修正した箇所については，「問題ないはずだ」という先入観でレビューを行い，問題点を見逃してしまう.	15	15	7	2	0		
9		代表制ヒューリスティック	・「この修正はUIの文言変更だけだから，サーバー側の処理に影響があるはずがない」と，典型的なイメージで判断し，実際にはその文言がログ出力や特定処理のキーになっている可能性を調査しない.	3	11	9	4	1	1.987	0.95
10			・「このバグは，前に経験したあの単純なバグと症状が似ている．原因も同じだろう」と決めつけ，より深刻な根本原因を見逃してしまう.	10	18	8	3	0		
11	自信と自己評価	ダニング＝クルーガー効果	・経験の浅いエンジニアが自身のスキルを過信し，「このくらいの実装なら，影響範囲は完全に把握できている」と判断してしまう.	20	14	4	1	0	1.654	0.86
12			・新しい技術要素について少し触っただけで全体を理解した気になり，複雑な依存関係を見落としてしまう.	22	11	4	1	1		
13		楽観主義バイアス	・「テストで問題が出なければ，本番でも問題は起きないだろう」と，予期せぬ本番環境固有の問題が発生する可能性を過小評価する.	12	13	8	6	0	2	1.013
14			・「何か問題があっても，すぐに修正（リカバリ）できるだろう」と，障害発生時の影響の大きさを楽観的に考えてしまう.	18	14	5	1	1		
15	社会的要因（集団心理と同調圧力）	同調圧力・バンドワゴン効果	・レビュー会議で他のメンバーが「問題ない」と言っているため，疑問に思う点があっても「自分が間違っているのかもしれない」と考え，指摘せずに流してしまう.	15	8	6	9	1	2.256	1.255
16			・チームリーダーや経験豊富なエンジニアの意見に，異論を唱えにくい雰囲気があり，潜在的なリスクが見過ごされる.	17	6	8	7	1		
17		権威バイアス	・技術顧問やアーキテクトが「その修正による影響はない」と言ったため，他のメンバーが疑問を感じつつも，それ以上の影響範囲の調査を行わなかった.	11	13	10	3	2	2.205	1.054
18			・経験豊富なベテランエンジニアが「自分の経験上，この変更で問題が起きたことはない」と発言したため，チーム全体がその意見を鵜呑みにしてしまい，潜在的なリスクが見逃された.	11	16	10	0	2		
19		集団浅慮（グループシンク）	・リリースが遅延している状況で，「これ以上問題点を指摘して，全体の和を乱したくない」という心理が働き，チーム全体としてリスクの高い判断を下してしまう.	15	12	7	4	1	1.705	0.921
20			・チーム内で「我々のチームは優秀だから大丈夫だ」という過信が生まれ，批判的な視点でのレビューが疎かになる.	26	13	0	0	0		
21		内集団バイアス	・「我々のチームのコードは品質が高いが，他チームが作ったモジュールは品質が低い可能性がある」と考え，自チームが担当する範囲の影響調査は甘くなり，他チームとの連携部分のリスクを見落とす.	26	12	1	0	0	1.462	0.674
22			・「以前，あのチームのレビューは形式的だったから，今回も深く考えなくていいだろう」と，特定のグループに対する先入観で対応を変えてしまう.	23	11	4	1	0		
23	現状維持と損失回避	現状維持バイアス	・「これまでこのやり方で問題なかったのだから，新しい影響範囲の特定手法を取り入れる必要はない」と，プロセスの改善を避けてしまう.	8	14	11	6	0	1.949	0.959
24			・影響範囲が広いと判明すると，修正作業が大規模になり大変だと感じ，「できれば影響範囲は小さいままであってほしい」と考え，調査を途中で打ち切ってしまう.	23	13	2	1	0		
25		損失回避性	・影響範囲を広めに特定することで，追加のテスト工数やスケジュール遅延が発生することを恐れ，意図的に影響範囲を狭く見積もってしまう.	23	10	4	2	0	1.603	0.882
26			・「もしここで新たな問題点を指摘したら，リリースが延期になって自分の評価が下がるかもしれない」と考え，リスクの報告を躊躇する.	24	10	2	3	0		
27		サンクコスト効果	・「この機能の開発には既に多くの時間を費やしてきた．今さら影響範囲が広いとは言えない」と考え，調査を簡略化してしまう.	22	11	3	3	0	1.974	0.96
28			・特定の技術的アプローチが問題だと分かりつつも，「ここまで作り込んでしまったのだから，後戻りはできない」と判断し，根本的な修正を避けてしまう.	9	13	14	3	0		

付録4_検証1 全員参加型のリスク洗い出し(チームA)

対象顧客	情報機器製品を利用するユーザー
開発プロダクト	情報機器制御用アプリケーション開発チーム
開発チーム	開発リーダー 1名、開発者 2名（中堅、若手）
対象要件概要	設定のエクスポート機能の仕様変更
リスクマトリクスの定義	発生確率、インパクト：未定義（チームで認識を合わせずに実施した）

＜個人によるリスク抽出結果＞

【開発リーダー】

設定ファイルを読み込む処理が初回起動時のフローに影響を与えかねない	初期値の内容が特定の機能に影響する	初期値の内容が他機能との整合性に影響を与えかねない

【開発者（中堅)】

設定ファイルの置き場が他製品とバッティングするリスク	設定値が他製品に影響を与えるリスク	他チームが実装した機能を流用できるのか

【開発者（若手】

既存製品へ本機能を展開した際に問題が起きかねない	ファイルパスを悪意あるパスに指定できてしまうのでは？	設定ファイルに悪意ある編集をされないか？

付録5_検証1 全員参加型のリスク洗い出し(チームB)

対象顧客	自動車メーカー向け(エンドユーザーは車を購入する個人)
開発プロダクト	自動車のコンポーネント
開発チーム	開発責任者(課長)1名、開発リーダー1名、開発担当者1名
対象要件概要	自社コンポーネントと繋がる顧客側製品とのプロトコル要件の変更に関して
リスクマトリクスの定義	チームで持ってるリスク分類の標準ルールに基づいて、検討した

＜個人によるリスク抽出結果＞

【開発責任者(課長】

エンコード方法を顧客と合意	変更可能性のある項目なきことを合意	閾値による影響を確認し問題なきことを確認

【開発リーダー】

重要データが他にないかを確認すること	誤って上位が別の信号を送ることがないかを確認すること	誤書き換えされる可能性がないかを確認すること

【開発担当者】

バージョンの際による送受信エラーがないかを確認すること	書き込み不要の情報を間違って書いたときに、どういう挙動になるかを確認すること

付録6_検証1 全員参加型のリスク洗い出し(チームC)

対象顧客	自動車メーカー向け(エンドユーザーは車を購入する個人)
開発プロダクト	自動車のモーター・ジェネレータ(略称:MG)
開発チーム	開発責任者(課長)1名、開発リーダー1名、開発担当者6名(協力会社2名含む)
対象要件概要	MG制御機能、故障診断通信機能、フェイルセーフ機能、イモビライザ機能、テスターを含むデータ通信機能のアップデート対応
リスクマトリクスの定義	発生確率：H/Wや外部要因依存度、知識・スキル依存度、過去プロジェクトの発生頻度や再現性で評価 インパクト：人命や安全性への影響、法規・ISO等の規格準拠度合い、影響機能数やQCDで評価

＜個人によるリスク抽出結果＞

【開発者A】

- ソフトウェア品番の重複リスク(他ECU品番と重複)
- ソフトウェア品番の指定間違い
(テスター側で対応のため対策不要)
- ソフトウェア品番の変換誤り(ASCIIコード⇔16進数変換ミス)
- ソフトウェア品番誤り防止のため、ドキュメントに対象品番を記入し、ツール上にも注意事項として記載する
- CAN送信側と受信側のデータ内容の不一致(分解能、MIN/MAX、単位、オフセット、初期値等)
- CAN通信データノイズによる仕様範囲外の値の送受信
(仕様範囲外の値は弾く、前回正常受信値を使う設計で対応済み)
- CAN通信変更する相手ECUのCAN通信管理表を顧客(自動車メーカー)に確認すること
- 相手ECUのCAN通信データ(Canoe/Canalyzer)を自動車メーカーから入手できるものはそれを使用する
- CAN通信環境構築時はCANデータの自動作成ツールを使用する
- SPI通信データ内容の同時性確保漏れ
(同一タイミングで送受信設計済み)
- SPI受信データが1回で全て受信できない場合は2回に分けて受信する
- テスター通信⇔MG間の通信タイムアウト時間は長いが、運用上問題ない(実害ない)ためケアしない
- CCP通信(RAMモニタ用)の処理負荷は考慮しない
(市場では機能しないデバッグ用のため)
- RAMモニタデータ追加に伴う処理負荷の超過(85%以上)がないか確認すること
- 処理負荷上限を80%以下⇒85%以下に緩和する

【開発者B】

- イモビライザ照合判定失敗時の動作制限内容(駆動禁止のみで良いか)を顧客に(自動車メーカー)に確認すること
- イモビライザ仕様とUDS仕様の整合性(DTCステータス)を自動車メーカーへ確認すること
- データフラッシュ追加データが多い場合は、空きセクタ(将来的な拡張用領域)の使用もOKとする
- データフラッシュの全書き込みデータが確定してからROM書き込みタイムアウト時間(10%マージン)を設定すること
- データフラッシュ書き換え回数上限(保証回数)突破による動作不良(15年以上は保証しない)
- 現開発フェーズではFFDデータの過不足は許容する
(次回以降のフェーズで適宜追加する)

【開発者C】

- 開発用鍵と量産用鍵の誤使用
- VC OFF(電源OFF)時の動作は保証しない
- 電源瞬断時の再初期化処理時間は許容する
(立上げ時間が少し遅れるのは仕方がない)
- 開発チームの時間的制約でレビュー時間を確保できない場合は書面レビュー(回覧レビュー)も可とする

【開発者D】

- B端子電圧の過電圧閾値(定格電圧)について、特許抵触リスクを確認すること
- 機能安全ASIL-Bを見たす範囲で、フェイルセーフ要件の緩和はOK
【SPFM(Single Point Fault Metric)】
ASIL-B ≥ 90%
【LFM(Latent Fault Metric)】
ASIL-B ≥ 60%
- ROM/RAM化けによる意図しない挙動発生
- 3重故障以上は保証しない
(2重故障までを保証する)
- 自動車メーカー任意設定可能なDTCは複数の故障を1つのDTCにグルーピング(緩和)してもOK

【開発者E】

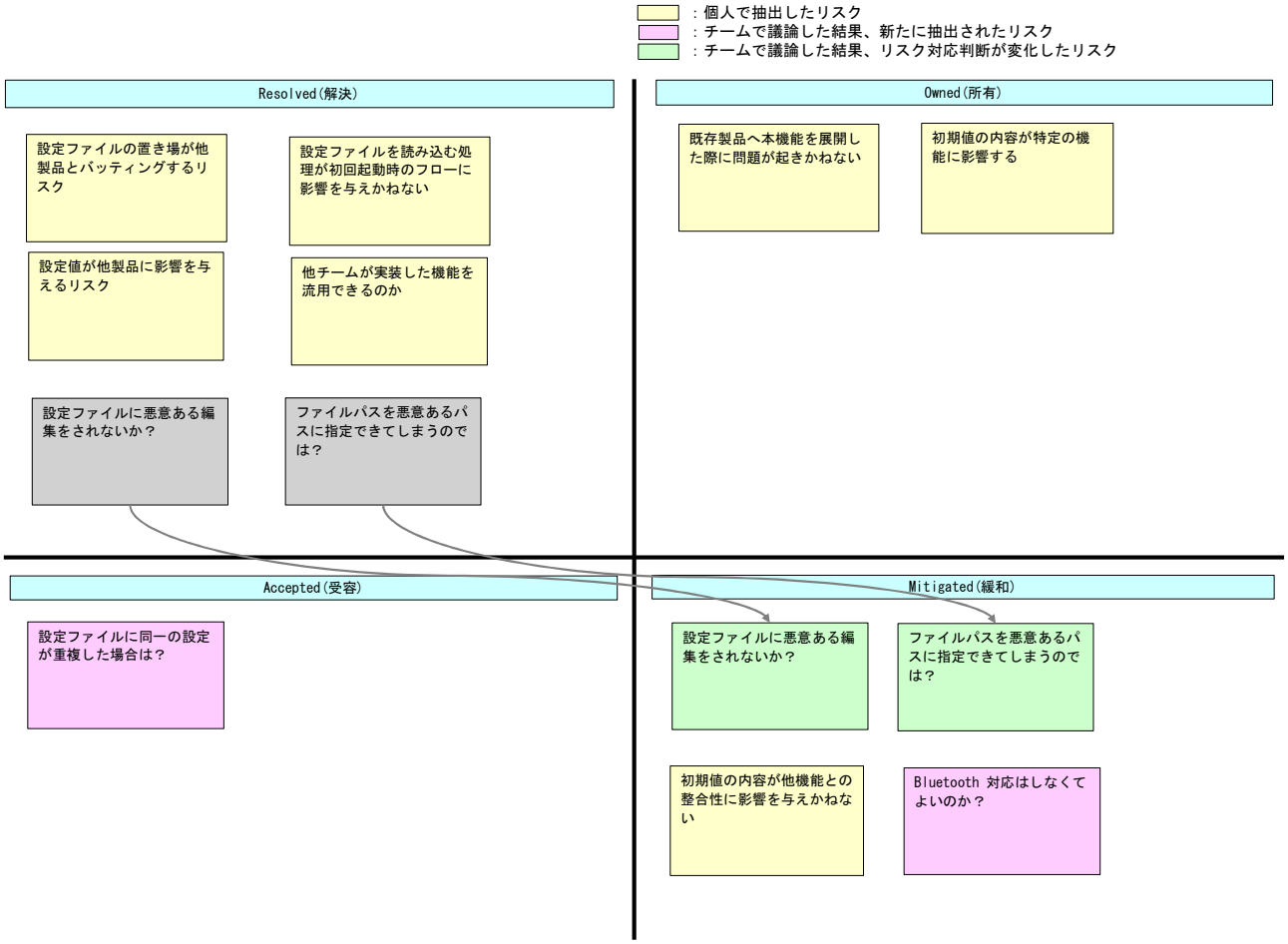
- 出力許容トルクの演算誤差は許容する
- 既存処理変更時のMISRA-C対応漏れ
- 駆動/発電許容回転速度閾値付近でチャタリング発生
(ヒステリシスを設けて対応済み)
- 変数の複数個所書き込み、複数回参照による意図しない動作(RMW)発生
- 変更に関わっている関数/変数/定数の抽出漏れ(Grep確認漏れ)
- MISRA-C(義務/必要/推奨)の推奨項目は正当な理由があれば緩和してよい

【開発者F】

- SID$14クリアデータの過剰消去、消去漏れ
- SID$22読み出しデータの過不足
- 試作品が完成次第、リプログラミングシーケンスを通すこと(変更後のシーケンスで正常動作することを確認すること)
- UDS規格(ISO14229)の逸脱
(フォーマット、応答優先順位等)
- 関数処理追加に伴うスタックオーバーフロー

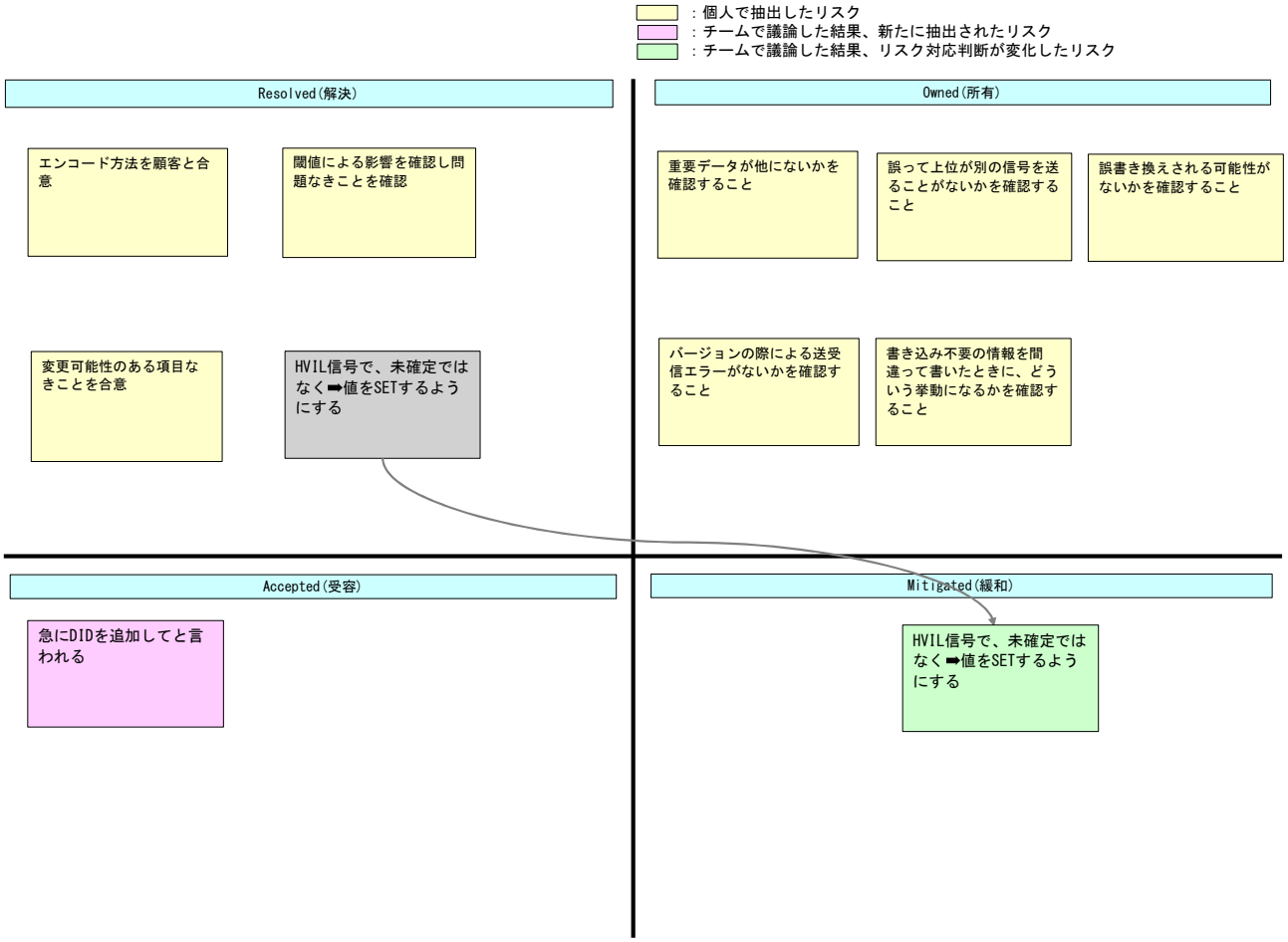
付録7_検証2 ROAMによるリスク分類結果(チームA)

個人で抽出したリスクをROAM分類した結果は以下の通り。
ROAM：リスクを Resolved（解決済み）、Owned（所有）、Accepted（受容）、Mitigated（緩和）の4つの対応方針に分類する管理手法。



付録8_検証2 ROAMによるリスク分類結果(チームB)

個人で抽出したリスクをROAM分類した結果は以下の通り。
ROAM：リスクを Resolved（解決済み）、Owned（所有）、Accepted（受容）、Mitigated（緩和）の4つの対応方針に分類する管理手法。



付録9_検証2 ROAMによるリスク分類結果(チームC)

個人で抽出したリスクをROAM分類した結果は以下の通り。
ROAM：リスクを Resolved（解決済み）、Owned（所有）、Accepted（受容）、Mitigated（緩和）の4つの対応方針に分類する管理手法。

：個人で抽出したリスク
：チームで議論した結果、新たに抽出されたリスク
：チームで議論した結果、リスク対応判断が変化したリスク

Resolved(解決)

ソフトウェア品番の重複リスク(他ECU品番と重複)	ソフトウェア品番の変換誤り(ASCIIコード⇔16進数変換ミス)	CAN送信側と受信側のデータ内容の不一致(分解能、MIN/MAX、単位、オフセット、初期値等)
CAN通信データノイズによる仕様範囲外の値の送受信(仕様範囲外の値は弾く、前回正常受信値を使う設計で対応済み)	SPI通信データ内容の同時性確保漏れ(同一タイミングで送受信設計済み)	ROM/RAM化けによる意図しない挙動発生
変更に関わっている関数/変数/定数の抽出漏れ(Grep確認漏れ)	UDS規格(ISO14229)の逸脱(フォーマット、応答優先順位等)	駆動/発電許容回転速度閾値付近でチャタリング発生(ヒステリシスを設けて対応済み)
関数処理追加に伴うスタックオーバーフロー	既存処理変更時のMISRA-C対応漏れ	変数の複数個所書き込み、複数回参照による意図しない動作(RMW)発生
SID$14クリアデータの過剰消去、消去漏れ	SID$22読み出しデータの過不足	開発用鍵と量産用鍵の誤使用

Owned(所有)

CAN通信変更する相手ECUのCAN通信管理表を顧客(自動車メーカー)に確認すること	RAMモニタデータ追加に伴う処理負荷の超過(85%以上)がないか確認すること	イモビライザ照合判定失敗時の動作制限内容(駆動禁止のみで良いか)を顧客に(自動車メーカー)に確認すること
イモビライザ仕様とUDS仕様の整合性(DTCステータス)を自動車メーカーへ確認すること	データフラッシュの全書き込みデータが確定してからROM書き込みタイムアウト時間(10%マージン)を設定すること	試作品が完成次第、リプログラミングシーケンスを通すこと(変更後のシーケンスで正常動作することを確認すること)
B端子電圧の過電圧閾値(定格電圧)について、特許抵触リスクを確認すること		

Accepted(受容)

データフラッシュ書き換え回数上限(保証回数)突破による動作不良(15年以上は保証しない)	ソフトウェア品番の指定間違い(テスター側で対応のため対策不要)	テスター通信⇔MG間の通信タイムアウト時間は長いが、運用上問題ない(実害ない)ためケアしない
CCP通信(RAMモニタ用)の処理負荷は考慮しない(市場では機能しないデバッグ用のため)	3重故障以上は保証しない(2重故障までを保証する)	VC OFF(電源OFF)時の動作は保証しない
出力許容トルクの演算誤差は許容する	現開発フェーズではFFDデータの過不足は許容する(次回以降のフェーズで適宜追加する)	電源瞬断時の再初期化処理時間は許容する(立上げ時間が少し遅れるのは仕方がない)

Mitigated(緩和)

ソフトウェア品番誤り防止のため、ドキュメントに対象品番を記入し、ツール上にも注意事項として記載する	相手ECUのCAN通信データ(Canoe/Canalyzer)を自動車メーカーから入手できるものはそれを使用する	CAN通信環境構築時はCANデータの自動作成ツールを使用する
SPI受信データが1回で全て受信できない場合は2回に分けて受信する	データフラッシュ追加データが多い場合は、空きセクタ(将来的な拡張用領域)の使用もOKとする	MISRA-C(義務/必要/推奨)の推奨項目は正当な理由があれば緩和してよい
機能安全ASIL-Bを見たす範囲で、フェイルセーフ要件の緩和はOK 【SPFM(Single Point Fault Metric)】 ASIL-B ≥ 90% 【LFM(Latent Fault Metric)】 ASIL-B ≥ 60%	自動車メーカー任意設定可能なDTCは複数の故障を1つのDTCにグルーピング(緩和)してもOK 開発チームの時間的制約でレビュー時間を確保できない場合は書面レビュー(回覧レビュー)も可とする	処理負荷上限を80%以下⇒85%以下に緩和する

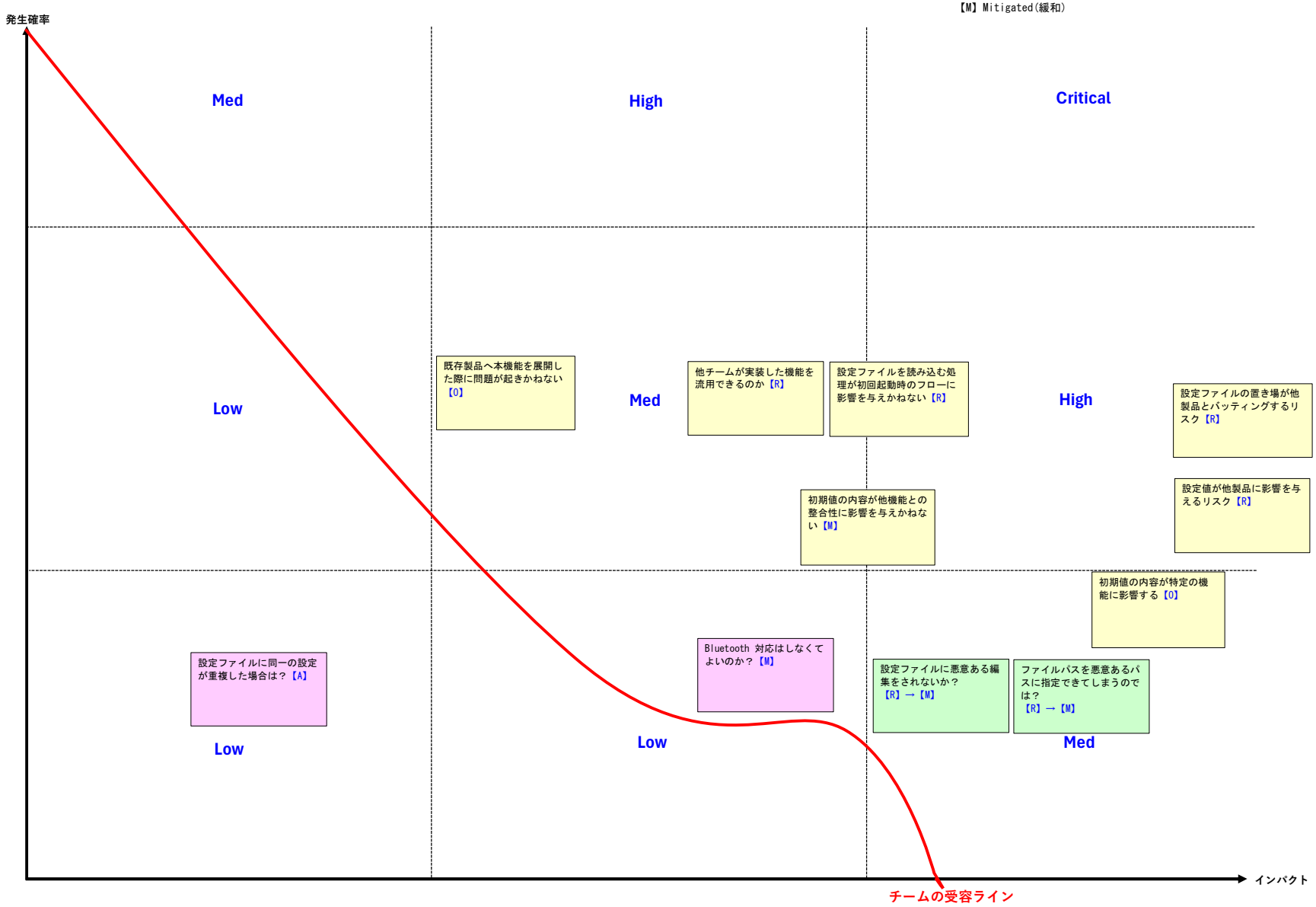
【補足事項】
本チームは、検証を実施する前から既にプロジェクトが開始(リスク抽出も完了)されていたこともあり、ワークショップ形式では実施できていません。
そのため、既に抽出済みのリスクをROAM分類した結果のみ記載している。
※「チームで議論した結果、新たに抽出されたリスク」及び「チームで議論した結果、リスク対応判断が変化したリスク」は上記に含まれていない。

付録10_検証2 リスクマトリクスによるチームの受容ライン決定(チームA)

ROAM分類したリスクを発生確率(縦軸)とインパクト(横軸)でマッピングし、チームの受容ラインを示したリスクマトリクス結果は以下の通り。

【R】Resolved(解決)
【O】Owned(所有)
【A】Accepted(受容)
【M】Mitigated(緩和)

【リスク評価】
Critical ＞ High ＞ Med ＞ Low

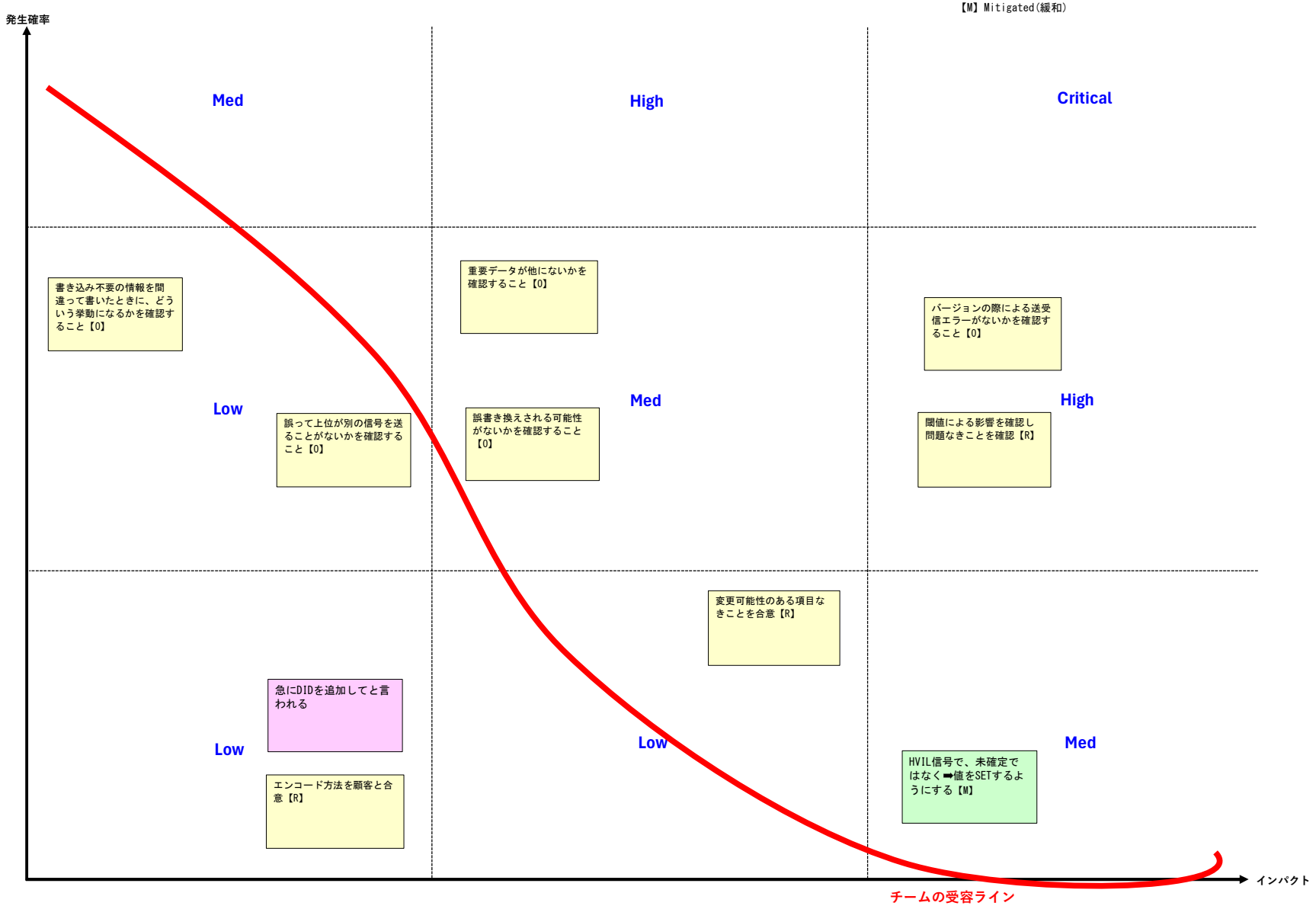


付録11_検証2 リスクマトリクスによるチームの受容ライン決定(チームB)

【R】Resolved(解決)
【O】Owned(所有)
【A】Accepted(受容)
【M】Mitigated(緩和)

【リスク評価】
Critical ＞ High ＞ Med ＞ Low

ROAM分類したリスクを発生確率(縦軸)とインパクト(横軸)でマッピングし、チームの受容ラインを示したリスクマトリクス結果は以下の通り。

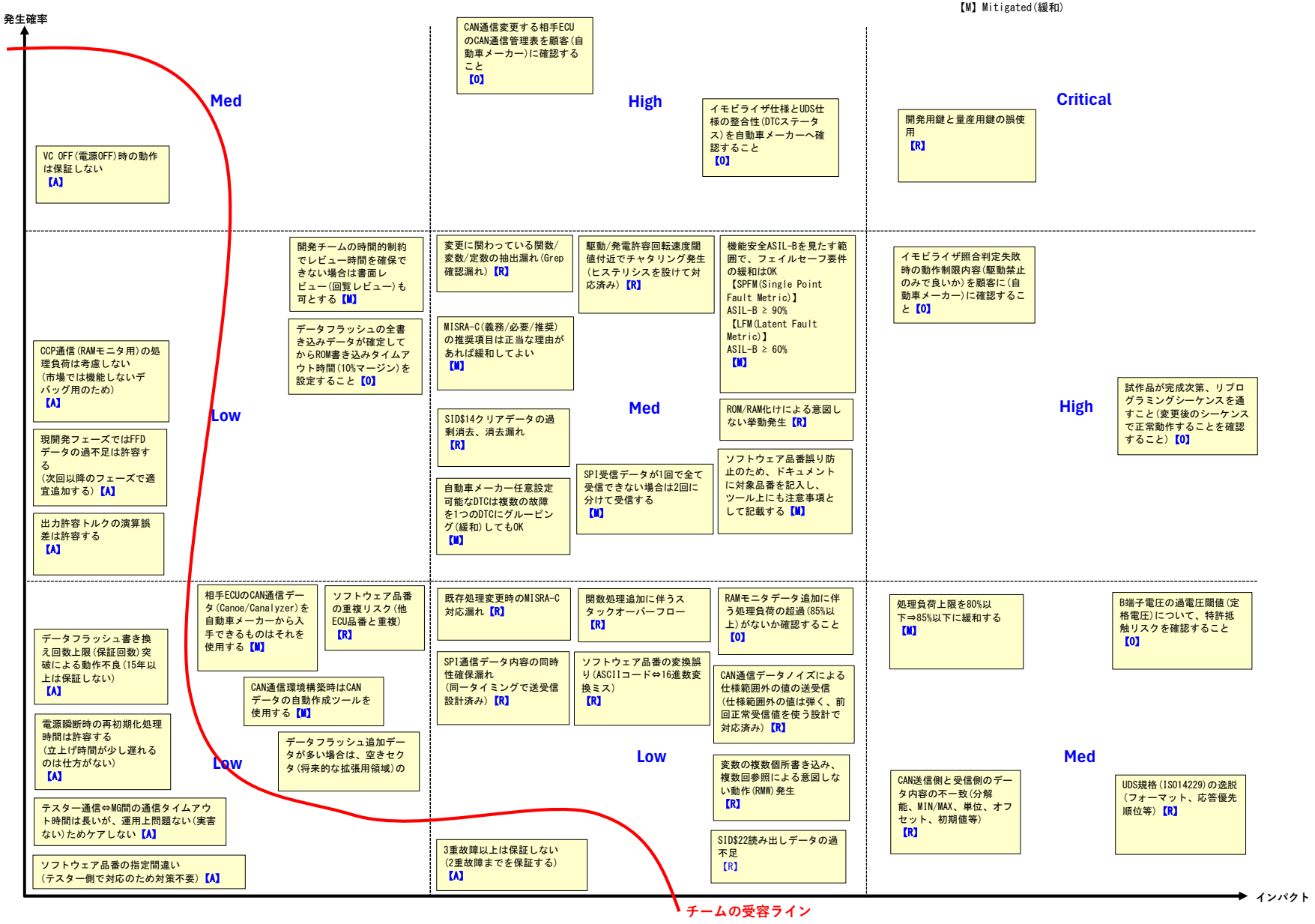


付録12_検証2 リスクマトリクスによるチームの受容ライン決定(チームC)

ROAM分類したリスクを発生確率(縦軸)とインパクト(横軸)でマッピングし、チームの受容ラインを示したリスクマトリクス結果は以下の通り。

【R】Resolved(解決)
【O】Owned(所有)
【A】Accepted(受容)
【M】Mitigated(緩和)

【リスク評価】
Critical ＞ High ＞ Med ＞ Low



付録13_検証3 持ち回り代表者によるリスク洗い出しとROAM分類(チームA)

対象顧客	情報機器製品を利用するユーザー
開発プロダクト	情報機器制御用アプリケーション開発チーム
開発チーム	開発リーダー 1名、開発者 2名（中堅、若手）
対象要件概要	ある特定の設定を追加する
リスクマトリクスの定義	発生確率：開発対象のソフトを使用したユーザーが問題に遭遇する確率 インパクト：即時対応が必要か（即時バージョンアップ or 次バージョンアップ時に対応 or 割切り）

＜代表者によるリスク抽出結果＞

※本検証では代表者を中堅の開発者にお願いをした

【開発者（中堅）】

- ある設定ごとにファイルを分ける必要があり、他機能に影響がある
- 媒体追加時に新たに追加したファイルの更新が必要となる
- ファームウェア側とのやり取りにより、特定機能に影響があるかも
- 新たな設定ファイル追加により、特定機能に支障がでる可能性あり
- 設定の保存場所に管理者権限が必要となる
- 設定値が同一製品の異なるモジュールに影響する
- ある表示機能に軽微な影響がある
- 設定反映にはアプリ再起動が必要となる
- ファームウェア側の特定の機能に影響がある

個人で抽出したリスクをROAMに分類した結果は以下の通り。

Resolved(解決)

- ある特定の設定ごとにファイルを分ける必要があり、他機能に影響がある
- 媒体追加時に新たに追加したファイルの更新が必要となる
- ファームウェア側とのやり取りにより、特定機能に影響があるかも
- 新たな設定ファイル追加により、特定機能に支障がでる可能性あり

Owned(所有)

- ファームウェア側の特定の機能に影響がある

Accepted(受容)

- 設定の保存場所に管理者権限が必要となる
- 設定値が同一製品の異なるモジュールに影響する
- ある表示機能に軽微な影響がある
- 設定反映にはアプリ再起動が必要となる

Mitigated(緩和)

付録14_持ち回り代表者によるリスク洗い出しとROAM分類のチームレビュー結果(チームA)

持ち回り代表者によるリスク洗い出しとROAM分類した内容を
チームでレビュー結果は以下の通り。

：個人で抽出したリスク
：チームで議論した結果、新たに抽出されたリスク
：チームで議論した結果、リスク対応判断が変化したリスク

Resolved(解決)

- ある特定の設定ごとにファイルを分ける必要があり、他機能に影響がある
- 媒体追加時に新たに追加したファイルの更新が必要となる
- ファームウェア側とのやり取りにより、特定機能に影響があるかも
- 新たな設定ファイル追加により、特定機能に支障がでる可能性あり
- 設定ファイルに失敗した時は？
→エラー処理を実装済み

Owned(所有)

- ファームウェア側の特定の機能に影響がある

Accepted(受容)

- 設定の保存場所に管理者権限が必要となる
- 設定値が同一製品の異なるモジュールに影響する
- ある表示機能に軽微な影響がある
- 設定反映にはアプリ再起動が必要となる

Mitigated(緩和)

- 特定の項目を追加した際に不整合は起きないか？

付録15_検証1,2 と 検証3 の実施時間比較

検証1, 2　　※開発メンバー3名 + 研究員1名

順番	手順	時間(分)	実施人数	合計時間(分)
1	対応中の変更のリスクを付箋で洗い出す。	7	4	28
2	ROAMの表に付箋を書いた人自身が最初にマッピングし、チームでそれが妥当か判断する。	15	4	60
3	次にリスクマトリクスにコピーした付箋をマッピングする。	12	4	48
4	リスクマトリクスを眺めて、不足が無いかを議論する	5	4	20
5	リスクマトリクスを眺めて、不足が無いかを議論する	5	4	20
			合計	**176 分**

検証3　　※開発メンバー3名 + 研究員1名

順番	手順	時間(分)	実施人数	合計時間(分)
(事前)	リスクの認識合わせ(発生確率、インパクトの基準を揃える)	3	4	12
(事前)	代表者がROAMの表に付箋を書いた人自身が最初にマッピングし、チームでそれが妥当か判断する。	15	1	15
1	次にリスクマトリクスにコピーした付箋をマッピングする。	23	4	92
			合計	**119 分**

削減時間割合　　**32.4 %**

【まとめ】
検証1,2 は176分かかったが、検証3では 119分 で完了した。
よって、**57 分**の時短ができ、**32.4%** 削減ができた。