付録1　プラクティスの肝 一覧

日本語名	「要約」	「効果」	予測	監視	対処	学習	技術スキル	非技術スキル	態度	心身の健康
イテレーション	プロジェクトで起こる変化（プロジェクトを開始する前の期待や想定とのギャップ）が大きく予想できない時には，一定期間を何度も繰り返して開発を進める．その結果，チームはプロダクト開発に必要な知識を学びながら，変化に適応していくことができる．	チームはプロダクトやプロジェクトについて獲得した知識や反省点を次のイテレーションに活かして改善することができる． その結果，プロダクトがステークホルダーの期待に近づく．イテレーションによる開発は，開発のリスクが影響する範囲を1イテレーションに納めることができる．それは，イテレーション毎に計画ミーティングやイテレーションの成果に対するレビューを行うためである．	・リスクに対しメンバーと繰り返し理解を深め，対応して進める． ・チームはプロダクトやプロジェクトについて獲得した知識や反省点を，次のイテレーションに活かして改善することができる．これを繰り返すことにより，計画の精度が向上し将来のリスクに備えるようになることができる．							
自己組織化チーム	チームメンバーが指示されて動くのではなく，各人の判断で動けるようにする．その結果，透明性の高い強い組織になる．	自ら判断しながら自律的にチームで仕事ができるようになる．PMの管理をするための工数が減り，常に最新の状況が可視化され透明性が増す．							・チームメンバーが指示されて動くのではなく，自分の判断で動くことが出来る	・メンバは，メンバが行なっている仕事の目的や重要性を理解している ・できた時は喜びあえる
迅速なフィードバック	アウトプットが適切かどうかわかりにくい時は，迅速なフィードバックを心掛ける．その結果，現状把握と軌道修正がしやすくなる．	プロダクトオーナーや顧客など情報を知りたい人が安心することができる． 日次ミーティングを1日に3回行うなど回数を増やすことによって，状況の把握に役立つようになった．								・メンバの行動や成果物に対し相互にフィードバックを適切に実施している ・問題が見つかってもすぐにフィードバックし安心出来る．
集団によるオーナーシップ	特定の人しか知らないことがないように，ソースコードや業務に関する知識を共同所有する．その結果，変化や問題に強いチームになる．	ペアプログラミングやソースコードレビューを通じて知識が共同所有されるようになり，さらにはチームの知恵が蓄積される．属人性が排除され，トラブルや変化に強いチームになる．			・特定の人しか知らないことが無いように業務の知識を共有すると他の人でも実施することができ，強いチームになる				・ワンチーム　困ったことがあっても，全員で知恵を出し合い，対応することで早期に解決する．	・メンバは，他のメンバが行なっている仕事の目的や重要性を理解している ・困ったことがあっても，咎めることは無い．全員で知恵を出し合い，対応するので安心し取り組むことができる．
アジャイルコーチ	もしチームが初めてアジャイル型開発に挑戦しようとしているなら，アジャイルコーチに導入を手伝ってもらうべきである．その結果，より確実にチームがアジャイルへの変化を体験できる．	チームがアジャイルのプラクティスを最短で学ぶことができる．プラクティスだけでなく背景にある考え方を学ぶことができる．チームが短期間で自立できるようになる．				・有識者や経験者に，直接指導を受けることで，プラクティスの教科書的な手法だけでなく，背景にある考え方を学習・理解することができる				
柔軟なプロセス	現時点でのプロセスが最善とは限らない．そのため，自らプロセスを改善していく．	組織が有する問題解決に適した納得の高いプロセスに成長してくる．			・誰かが決めたやり方ではなく，現場の問題が解決できる様にプロセスを改善していく			・状況や目的に応じて適切なプロセスを検討・実行できる．		
ふりかえり	最初から開発プロジェクトに最適な手法がわからないのであれば，イテレーション毎にふりかえり，学んだことを共有してより最適なやり方を編み出す．その結果チームの現状に最適な開発プロセスを作りあげていくことができる．	イテレーション毎にチームの開発プロセスや規律を改善できる．早期の比較的小さい失敗から学ぶことができ，大きな失敗をしにくくなる．チームで学習する結果，個々のメンバーの成長が早くなる．				・開発途中で生じた事柄をチームで共有し，特に良い事については，他のメンバも実践するようにしている				
ペアプログラミング	チームの中で知識やコードの共有ができていない場合は，ペアを組んで作業をする．その結果，開発チームの中で業務知識やコードについての知識が共有でき，品質や作業効率も向上できる．	作業の結果についての質が高まりやすい．難しい問題が早く解決しやすくなる．作業やコードの内容についての理解がチームに広まる．規律を守りやすくなる．開発が楽しくなる．			・開発チームの中で業務知識やコードについての知識が共有でき，理解がチームに広まることで大きな問題が発生しても早く対応できる	・開発チームの中で業務知識やコードについての知識を効率よく共有・学習することができる．	・専門知識がもつメンバーのスキルを短期間でメンバーに展開できる． ・複数人で共に作業を行うことで，個々の知らないことを教えあい，業務知識を効率よく共有することができる．			
人材のローテーション	属人化しているならば様々な役割が経験できるようにする．その結果，変化やインシデントに強い組織になる．	ファシリテータやスクラムマスターなどの役割についても，ローテーションすることによって，その役割についての共通理解が広がる．属人性が減少し，変化やインシデントに対する耐性が強い組織になる．共通理解が広がると，ふりかえりやレビューなどで，より成熟した解決策を選択し，実施することができる．	・プロジェクト計画時に特定の人しかできない業務があれば，他の人でも対応できる様に計画を作成する． ・チームメンバーが多様な業務技術や役割を学ぶことで属人性が減少し，変化やインシデントに対しての予見能力が強い組織になる．		・開発チームの中で専門技術や役割が共有できているので，大きな問題が発生しても早く対応できる	・業務技術や役割を学ぶことができ，知識の共通理解が広がる．	・属人化しない様に，メンバーが持つ技術スキルをドキュメント化したり，学習や教育できる様に積極的にすすめる． ・業務内容や役割に応じた技術スキルを学ぶことができ，知識の共通理解が広がる．			
イテレーション計画ミーティング	イテレーションを始める際に，イテレーションで達成すべきゴールと，それを実現する作業を洗い出そう．その結果，チームが開発を進めるために必要かつ具体的な計画を立てることができる．	イテレーションで達成できるユーザーストーリーと，それを完了するために必要なタスクと作業時間を見積もることができる．チームで作業を洗い出すため，そのイテレーションで何をしなければいないかを全員が共有できる．	・問題が発生しても，イテレーションは短いので，その期間内でできる現実的で具体的な計画を無理なく着実に立案することができる． ・イテレーション期間内での具体的な作業内容をチーム全員で検討することで，先のリスク含む高精度の計画を立てることができる．							
プランニングポーカー	開発規模の見積り精度が個人のスキルに依存する，非作業者が見積っている場合は，チーム全員で素早く見積もっていこう．その結果，全員の知識や経験を活かした見積り結果を得ることができる．	チームの持っている知識や経験をいかした，確度の高い見積りができる．見積り根拠を互いに述べ合うことで，開発対象や見積りに対する理解が深まる．ゲーム感覚で活発な対話を引き出し，チームの相互理解を深める．	・チーム全員の知識や経験を生かした見積もりを素早く作成できる． ・見積り作業を全員で行うことで，メンバーの知識や経験を活かした確度の高い見積り結果を得ることができる．							
スプリントレビュー	イテレーションの終わりに完了したものを関係者にデモをする．その結果，チームは次のイテレーションで何をするべきかフィードバックを得る．	イテレーションの成果をステークホルダー全員で共有することができる．レビューの結果を元に必要に応じてプロダクトバックログを変更し，次のイテレーションでやるべきことを検討することができる．				・チームの成果物をステークホルダー全員で共有し，良い結果を次のイテレーションへ反映する．				

日本語名	「要約」	「効果」	予測	監視	対処	学習	技術スキル	非技術スキル	態度	心身の健康
ベロシティ計測	今後のチームの作業量を予測したいならば，チームのイテレーションあたりの開発量を計測し続ける．その結果，ある期間までの開発量の予測や，現在の開発規模についての着地予測を見積もることができる．	チームのベロシティを元にリリース計画を立てることができる．状況に応じたリリース計画を立て続けることができる．	・過去の開発速度から，問題が発生しても現実的な計画を精度高く作成できる． ・チームのベロシティ値が存在することでそれを根拠として，開発量の予測や現在の開発規模についての着地予測を見積もることができる．							
リリース計画ミーティング	いつどの機能をリリースするかについての計画を立てるミーティングを開く．その結果，チームとプロダクトの関係者が共通の目標を持つことができる．	リリース計画によっていつどの機能を提供するかという，日付入りの目標を持つことができる．利用価値の高い機能を優先してリリースすることによって，早期に投資の回収が行える．プロダクトのリリース計画を踏まえた組織の戦略を立てることができる．	・問題が発生しても，顧客や開発メンバーとリスクを含めて話し合い，共通のゴールに向け計画を作成している． ・チームとプロダクトの関係者が共通の目標を持つことができるため，最終的なゴールに向けての適切な作業内容を検討することができる．							
日次ミーティング	毎日，時間を決めて短い時間で関係者が顔を合わせる．その結果，チーム全体が日々の必要な情報を共有できるようになる．	情報共有の漏れを防ぎ，問題の把握と対処を適切なタイミングで行うことができる．朝の日次ミーティングはチームの1日のリズムを整える時間となる．	・急にリスクが発覚しても，問題が大きくなる前にすぐにメンバー全員と共有・理解した上で，適切に対応できる． ・日単位でチーム全員が状況を共有・理解することにより，リスク発生などの確認を密に行うことができる．							
かんばん	プロジェクトを取り巻く状況が不安定で計画を頻繁に変更せざるを得ない場合には，同時に開発する機能の数を制限して流量コントロールする．その結果，機能を流れるように提供することができ変化に対応しやすい．	あらかじめ計画を立てなくても効率的に開発を進めることできる自然に障害を取り除くようになり作業の流れを最適化しやすい．作業の流れを最適化しようとすることで協働作業が促進される．		・メンバの負荷状況やプロジェクトの進捗状況が見える化できており，異常な状態であると判断するメトリクスを設定している ・作業流れを見える化し担当者しかわからない異常な状況を作らない様にする				・作業流れを見える化し異常な状況が発生する前にメンバー間で情報共有を進め，調整し合うことができる		
タスクボード（タスクカード）	チームのタスク状況を可視化するために，タスクカードを用意し，状態を管理するタスクボードにはりつける．その結果，だれがどのタスクを担当しているか，どこかに異常が起きていないかなどをチーム全員が把握できるようになる．	タスクの見える化によって，チームメンバーの異常を検知できるようになる．その結果，チームで相互協力しやすい．タスクの実施漏れを防ぐことができる．チームの状況が表現されているため，それを見ながら日次ミーティングを実施しやすい		・メンバの負荷状況やプロジェクトの進捗状況が見える化できており，異常な状態であると判断するメトリクスを設定している ・担当者の作業をタスクにして見える化し異常な状況を作らない様にする						
スプリントバックログ	イテレーションで何をするべきか，スプリントバックログにリストアップする．イテレーションの残タスクを追跡することによって，進捗を把握することができる．	チームが，イテレーションで何を開発するべきか，タスクとして何が必要かを明確に理解できるようになる． タスクの残作業時間を計測するため，イテレーションの進捗を管理することができる．		・メンバの負荷状況やプロジェクトの進捗状況が見える化できており，異常な状態であると判断するメトリクスを設定している ・チームの作業を見える化しリスクを考慮し異常な状況を作らない様にする						
インセプションデッキ	プロダクトの目的や方向性が曖昧のまま開発を進めている場合は，プロダクトの目的や方向性を明らかにした10の質問に回答しチームで共有する．その結果，チームはプロダクトの目的，ビジョン，方向性を理解して開発が進めやすくなる．	プロジェクトの目的や背景についての方向性がステークホルダーの間で合致し，チームが状況に応じた適切な判断を下せるようになる．ステークホルダーにプロジェクトについての情報を提供することになり，ステークホルダーがプロジェクトを続けるかどうかの判断を下せるようになる								・メンバは，他のメンバが行なっている仕事の目的や重要性を理解している ・メンバー全員がプロジェクトの目的や重要性を理解しやりがいを持って業務をすすめることができる
プロダクトバックログ（優先順位付け）	プロダクトとして何を作るべきかをプロダクトバックログで管理し，その優先順位をチームとの会話を踏まえ，プロダクトオーナーに判断してもらう．その結果，チームは何から開発を行えばよいか明確になる．	開発チームが，何をどこから開発するべきか明確に理解できるようになる． プロダクトにとって何が重要なのかが順序づけされ理解しやすい． 変化が発生しても，スプリントごとに再度プロダクトバックログを見直し適応出来るようにする	・最悪な場合でも，成功できる様にリスクの高いタスクを優先して対応し異常な状況を作らない様に計画する． ・プロジェクトにとっての優先順位が共有されることで，先の工程まで効率的に作業を進めることができる．		・最悪な場合が発生しても，成功できる様に優先順位，プロセス，計画を見直し異常な状況を作らない		・新規技術スキルの習得や調査も優先度のある作業として，計画に含めることができる．			
自動化された回帰テスト	システムを修正する度に回帰テストを手動で実施しなければいけないなら，回帰テストを自動化すること．その結果，回帰テストにかかる工数が大幅に削減でき，何度も実施できるようになる．	回帰テストの実施時間を大幅短縮できる．既存の機能の不具合を早期発見できる．								・メンバは有意義な作業のみを実施し，退屈で繰り返し実施する作業を極力減らす努力を行っている．
リファクタリング	コードがわかりにくい，複雑である，汚いと感じたら，コードの振る舞いは変えずに内部の設計を改善する．その結果，コードの見通しがよくなり，将来的な設計へのリスクを軽減することができる．	アーキテクチャも含め，より設計改善を行うことができる． メンテナンスしやすいコードを維持することができる． リファクタリングは実施しているが，内部構造の改善につながっているかどうかは定かではない場合もある．					・定期的なリファクタリングを実施してソースコードのメンテナンス性を維持することで，技術的負債が低減することができる．			
継続的インテグレーション	インテグレーション（システムのビルド，テストの実行）を自動化し，継続的に行うことによって，コードだけでなく動作環境を含めた確認を行うことができる．	継続的インテグレーションを行うことで，継続的に「出荷できる状態」にできる． 新しく追加されたコードが問題既存のバグ を引き起こしていても，早期に検知できる． 問題に早期に対処することで継続的に出荷可能な状態を保つことができる． 動作環境も含め，継続的インテレーションできるため，自動テストやアジャイル型開発の基盤の一つとなる．					・継続的インテグレーションを導入することにより，特定の領域(システムビルドなど)において個々の技術スキルに依存しない品質を担保することができる．			・継続的インテグレーションを導入することで，リリース等の心身に高いストレスがかかる作業を軽減する努力を行っている

日本語名	「要約」	「効果」	予測	監視	対処	学習	技術スキル	非技術スキル	態度	心身の健康
コーディング規約	チームでの開発を始める時，メンバーで合意形成をしながらコーディング規約を決める．その結果，チームの合意形成の練習になり，コードの統一性が増して可読性が高まる．	ツールでコーディング規約を強制することで，統一的なプログラムコードになる．ソースコードの可読性が高まる．チーム内の合意形成と，コミットメントに従わないメンバーに対する許容についての練習になる．			・ソースコードの可読性を高めることで，コード上で行われている処理に対し認識をメンバー間で共有できる．					
紙・手書きツール	チームの中で何かを議論したり，共有したりする場合には，付箋やホワイトボードを充分に活用しよう．	チームは，現在の状況をいつでも見ることができる．様々な変更や修正を素早く他の人に伝え説明でき，誤解や行き違いの発生を防ぐ．付箋を使わない場合は，管理ツール上の情報と現状との差異が発生していたが，付箋を使うことによって，最新の状況が表現できる．紙・手書きツールは柔軟であるため，現場の創意工夫を活かしやすい．ただし，分散拠点で利用しようとするのは難しい．			・議論の際，手軽に意見や情報をアウトプットできるツールを用いることで，認識の共有を容易に行うことができる．			・議論の際，手軽に意見や情報をアウトプットできるツールを用いることで，相手に伝わりやすくする		
オンサイト顧客	ムダなソフトウェア機能が作られてしまうことを防ぐため，開発者と顧客が常に会話ができる環境で仕事をしよう．その結果，顧客と開発者のコミュニケーション量が増えて，顧客が望むソフトウェアが作られやすくなる．	顧客は開発しているソフトウェアをいつでも見ることができる．ビジネス上の変更を素早く開発者に伝え説明でき，開発者との誤解や行き違いの発生を防ぐ． 開発者は疑問点を相談し，仕様上の意志決定を顧客に素早く依頼できる．作ったもののフィードバックを得られやすくなり，ムダな機能が作り込まれることを防ぐ．同じ場所で働くことで，開発者と顧客の間に信頼関係が生まれ，小さな相談も気軽にできるようになる．							・絶えず顧客と相談することで無駄な機能を作らないようにしている	
プロダクトオーナー	要件の優先順位や仕様がなかなか決まらない場合は，プロダクトの結果に責任を持ち優先順位や仕様を決める役割を設ける．その結果，プロダクトについての決定を素早く行うことができる．	プロダクトのコンセプトに沿う形で要件をまとめ，素早く意思決定することができる．開発者と密にコミュニケーションを取ることで誤解が生じにくい	・チームへの作業指示が一本化され，不要・不明確な作業を減らすことができるため作業予測が立てやすくなる．		・優先順位や仕様を決める役割を設けることで，プロダクトに関する決定を素早く行うことができる．					
ファシリテータ（スクラムマスター）	開発チームの内外に調整が必要であれば，ファシリテータの役割を設置する．その結果，開発が進みやすくなる	開発チームの活気を持続することができる．プロダクトオーナーやチームの外部のステークホルダーとの調整が行える．						・第三者の視点を持つことで，冷静にチーム内の問題を解決できる		
ニコニコカレンダー	カレンダーに毎日その日の気持ちを表すシールを貼るようにする．その結果，チームのムードやメンバーの気持ちが見えるようになる．	「自分の気持ちを表現してよい」という場を作り，チームの絆を深める．気付きにくいメンバーの状態を，チームお互いで把握できるようになる．メンバーの状態に対して，適切なタイミングで対処を行うことができる．		・メンバが報告しやすい状況・環境が整っている						・メンバーの精神状態を，明示的に確認できる手段が存在することで，異常を察知し，適切なタイミングでケアを行うことができる．
持続可能なペース	高い効果や生産性を維持できるよう持続可能なペースを保つ．その結果，健康的で活気のある職場になる．	メリハリのある仕事のリズムを刻めるようになる．一気に作りあげて，その後が続かないのではなく，継続的に安定的な効果を提供することができる．								・無理して後が続かないやり方でなく，健康的でメリハリのある活動を心がけている．また，メンバーの一人一人が，安心して自分らしく働くことができる．
共通の部屋	意思疎通がうまくいかず，活気がない場合は，共通の部屋を作る．その結果，コミュニケーションは改善する．	チームに活気が出て，コミュニケーションが円滑になる．お互いの信頼関係を醸成する土壌となる．			・物理的な距離が近くなることで協力しやすい状況になる．			・物理的な距離が近くなることで円滑にコミュニケーションが取れることを理解し実践出来る．		
チーム全体が一つに	チーム全員が一つの目標に向かうような取り組みを行う．その結果，チームは素早く正確にコミュニケーションができ，生き生きとした仕事ができるようになる．	チームメンバーそれぞれが仕事の目標を捉えることで，ゴールに向かった振る舞いを取ることができる． 行き詰ったら，他のメンバーに助けを求められるようになる．コミュニケーションが効果的になる．推測するのを止めて，お互いに質問や相談をするようになる．アイデアが伝達されやすくなる．チームに一体感と互いへの敬意が生まれる．仕事が楽しくなる．			・チームメンバーそれぞれが仕事の目標を捉えることで，ゴールに向かった適切な振る舞いを取ることができる．		・メンバー各自が保有する技術スキルを共有することで効率よくスキルの理解や継承を行うことができ，チーム全体のスキルレベルが向上しやすくなる．	・積極的な質問や相談が行われることで，素早く正確にコミュニケーションができる．	・チームメンバ全員が一つの目標に向かった振る舞いを取ることが出来る	
心理的安全	他者からの反応に怯えたり，羞恥心を感じたりすることなく，自然体の自分自身を晒せる環境	一人ひとりが本来の自分を曝け出すことができること，そして，それを受け入れるための「心理的安全性」，つまり他者への心遣いや共感，理解力を醸成することが，間接的にではあるが，チームの生産性を高めることにつながる．		・メンバが報告しやすい状況・環境が整っている ・実現性や不明点のリスクなどを事前に把握し，チームで共有している						

「心理的安全」以外のプラクティス 出典:「IPA アジャイル型開 https://www.ipa.go.jp/sec/softwareengineering/reports/20130319.html
「心理的安全」プラクティス 出典：「グーグルが突きとめた！社 https://gendai.ismedia.jp/articles/-/48137

付録2 アンケート調査票

分類	No	質問	【必須】チームメンバの人数	【必須】開発スタイル （記入例：該当する場合、□→■にする）
チーム状況把握	1	アンケート対象のプロジェクトチームについて教えてください		□アジャイル型 □ウォーターフォール型 □その他（　　）

7. 強く当てはまる　3. やや当てはまらない
6. 当てはまる　2. 当てはまらない
5. やや当てはまる　1. まったく当てはまらない
4. どちらともいえない

分類		No.	質問	【必須】回答欄 (数値1～7を記入)	【必須】実践しているアジャイルプラクティス （記入例：該当する場合、□→■にする）	【任意】コメント欄 （他に実践しているプラクティスなど）
基本能力	予測	1	プロジェクト計画時、顧客やメンバと話し合い優先順位とリリース時期を明確にした上で計画を作成している 対人的・時間的リソースにリスクは無いか？特定の人しかできない業務はないか？といったことまで考慮している		□リリース計画ミーティング □プロダクトバックログ（優先順位付け） □プロダクトオーナー □人材のローテーション	
		2	プロジェクト中にリスクが発覚した場合、具体的な内容をチームメンバ全員が共有・理解した上で作業を進めている 例：実現性や不明点のリスクなどを事前に把握し、チームで共有している		□イテレーション計画ミーティング □イテレーション □日次ミーティング	
		3	要件変更があった場合、計画変更を行いチームとして対応が可能な範囲を再定義している 例：人的・時間的リソースや負荷状況をチームで共有した上での、実現性のある計画を立てている		□プランニングポーカー □イテレーション計画ミーティング □イテレーション □ベロシティ計測	
	監視	4	メンバの負荷状況やプロジェクトの進捗状況が見える化できており、異常な状態であると判断するメトリクスを設定している		□かんばん □タスクボード □スプリントバックログ	
		5	メンバが報告しやすい状況・環境が整っている		□ニコニコカレンダー □心理的安全	
		6	深刻な問題を招く恐れのある状況が見落とされることなく、問題として認識できる		□日次ミーティング	
	対処	7	メンバーが急に減った(例：入院した)場合、そのメンバーの担当していた作業を短期間で引き継げる		□ペアプログラミング □コーディング規約 □人材のローテーション	
		8	問題の解決のためにメンバ間の作業調整を行う際、特定のメンバに作業が集中しないように分担できている		□柔軟なプロセス □集団によるオーナーシップ	
		9	解決が困難な問題が発生した場合、問題の解決のためにメンバー同士が協力して対応できる		□紙・手書きツール □チーム全体が一つに □共通の部屋	
		10	影響の大きな設計変更や機能追加が必要となった場合、柔軟に計画見直しやプロセスの見直しができる		□プロダクトオーナー □プロダクトバックログ（優先順位付け）	
	学習	11	スキルの高いメンバーが、低いメンバーに対して、日常的に指導している		□ペアプログラミング □スプリントレビュー	
		12	開発途中で、今までの作業の振り返りを行い、良い結果をチーム内で共有し、他のメンバも実践している		□ふりかえり	
		13	新たな開発手法を取り入れた時、その開発手法の有識者や経験者に直接指導を受けている		□アジャイルコーチ	
		14	会社やチームがメンバの学習する環境を積極的に提供している		□人材のローテーション	

分類		No.	質問	【必須】回答欄 (数値1～7を記入)	【必須】実践しているアジャイルプラクティス （記入例：該当する場合、□→■にする）	【任意】コメント欄
レジリエンス能力を支える要素	技術スキル	15	開発メンバが保有する固有の技術スキルについて、メンバ間での共有(学習・教育含む)が適宜行われている		□人材のローテーション □チーム全体が一つに □ペアプログラミング	
		16	技術スキルが不足しているメンバでも作業が滞りなく実施できるよう、開発作業に必要となる情報はドキュメント化がされている状態である		□人材のローテーション □リファクタリング	
		17	将来の開発において必要となりうる技術スキルに関し、チームとして習得に向けた活動(勉強会など)を作業として計画している		□プロダクトバックログ □継続的インテグレーション	
	非技術スキル	18	メンバ間での情報共有が活発に行えている		□紙・手書きツール □共通の部屋 □かんばん	
		19	問題解決する際、複数の解決案から適切なものを選択できている		□柔軟なプロセス	
		20	チーム内で意見が分かれた場合、それぞれの主張をお互い理解してから解決するようにしている		□チーム全体が一つに □ファシリテータ（スクラムマスター）	
	態度	21	特定の人しか知らないことがないように、ソースコードや業務に関する知識を共有している		□集団によるオーナーシップ	
		22	チームのメンバは、身勝手な判断や行動ではなく、各自の判断で自律的に動いている		□自己組織化チーム	
		23	チームメンバ全員が一つの目標に向かった振る舞いを取っている		□チーム全体が一つに	
		24	絶えず顧客と相談することで無駄な機能を作らないようにしている		□オンサイト顧客	
	心身の健康	25	メンバの行動や成果物に対して、メンバが相互にフィードバックを適切に実施している		□迅速なフィードバック	
		26	メンバは、自分や他のメンバが行なっている仕事の目的や重要性を理解している		□インセプションデッキ □集団によるオーナーシップ □自己組織化チーム	
		27	メンバは有意義な作業のみを実施し、退屈で繰り返し実施する作業はしなくても良い		□自動化された回帰テスト □継続的インテグレーション	
		28	無理してあとが続かないやり方でなく、健康的でメリハリのある活動を心がけている。また、メンバーの一人一人が安心して自分らしく働いている		□ニコニコカレンダー □持続可能なペース	

付録3-1 アンケート結果（レジリエンスに対する自己評価）

分類	No.	質問	平均	チーム名																										
				AG_A	AG_B	AG_C	AG_D	AG_E	AG_F	AG_G	AG_H	AG_I	AG_J	AG_K	AG_L	AG_M	WF_A	WF_B	WF_C	WF_D	WF_E	WF_F	WF_G	WF_H	WF_I	WF_J	HY_A	HY_B
予測	1	プロジェクト計画時、顧客やメンバと話し合い優先順位とリリース時期を明確にした上で計画を作成している	5.60	6	6	6	5	6	6	6	6	5	5	5	5	5	6	5	5	5	5	4	6	6	5	7	7	7
	2	プロジェクト中にリスクが発覚した場合、具体的な内容をチームメンバ(全員が共有・理解した上で作業を進めている	5.56	5	5	4	4	7	6	6	7	6	6	4	5	7	5	5	4	5	6	7	5	6	4	7	7	6
	3	要件変更があった場合、計画変更を行いチームとして対応が可能な範囲を再定義している	5.28	6	5	5	4	6	6	4	5	7	6	6	6	7	6	5	5	4	6	7	5	3	5	1	7	5
監視	4	メンバの負荷状況やプロジェクトの進捗状況が見える化できており、異常な状態であると判断するメトリクスを設定している	4.96	5	6	5	5	6	5	6	6	6	3	3	5	6	5	4	5	5	4	7	5	3	5	1	7	6
	5	メンバが報告しやすい状況・環境が整っている	4.96	5	6	5	3	6	6	3	5	4	5	6	6	7	5	6	5	4	3	7	5	3	4	7	7	1
	6	深刻な問題を招く恐れのある状況が見落とされることなく、問題として認識できる	5.32	6	4	5	5	5	5	6	5	2	5	6	6	7	6	6	4	5	6	7	5	3	4	7	7	6
対処	7	メンバーが急に減った(例：入院した)場合、そのメンバーの担当していた作業を短期間で引き継げる	4.60	3	4	4	6	4	2	5	6	2	6	5	4	7	5	7	3	4	5	6	5	4	4	1	6	7
	8	問題の解決のためにメンバ間の作業調整を行う際、特定のメンバに作業が集中しないように分担できている	4.32	5	5	4	3	4	5	6	6	2	4	3	4	7	3	6	3	5	5	2	5	3	4	1	6	7
	9	解決が困難な問題が発生した場合、問題の解決のためにメンバー同士が協力して対応できる	5.76	5	7	5	6	7	7	6	7	4	6	6	7	7	5	5	4	5	5	7	5	4	5	7	7	5
	10	影響の大きな設計変更や機能追加が必要となった場合、柔軟に計画見直しやプロセスの見直しができる	5.52	4	6	5	5	7	6	5	5	7	6	3	7	7	6	5	5	5	6	4	5	4	6	5	7	7
学習	11	スキルの高いメンバーが、低いメンバーに対して、日常的に指導している	4.56	2	5	4	6	6	2	4	7	3	4	4	5	6	6	6	2	5	4	7	5	3	5	1	7	5
	12	開発途中で、今までの作業の振り返りを行い、良い結果をチーム内で共有し、他のメンバも実践している	4.36	5	7	3	1	6	2	3	6	1	7	4	5	7	3	5	4	2	5	5	5	4	5	1	7	6
	13	新たな開発手法を取り入れた時、その開発手法の有識者や経験者に直接指導を受けている	3.68	6	4	2	4	4	4	2	5	1	5	1	5	2	3	4	5	3	3	5	4	4	4	1	7	4
	14	会社やチームがメンバの学習する環境を積極的に提供している	4.84	6	5	3	4	5	3	6	6	7	5	3	6	5	3	3	5	5	3	7	4	5	3	7	7	5
技術スキル	15	開発メンバが保有する固有の技術スキルについて、メンバ間での共有(学習・教育含む)が適宜行われている	4.52	5	5	2	5	6	3	7	6	3	5	4	6	5	6	5	3	5	3	5	4	4	3	1	7	5
	16	技術スキルが不足しているメンバでも作業が滞りなく実施できるよう、開発作業に必要となる情報はドキュメント化がされている状態である	4.52	2	6	5	5	3	3	3	5	4	5	3	7	6	3	4	3	3	4	6	5	5	5	5	7	6
	17	将来の開発において必要となりうる技術スキルに関し、チームとして習得に向けた活動(勉強会など)を作業として計画している	3.96	6	2	3	2	7	2	7	5	2	3	4	7	6	2	3	3	2	3	6	5	4	2	1	7	5
非技術スキル	18	メンバ間での情報共有が活発に行えている	5.08	4	5	5	5	7	6	6	6	3	5	4	6	7	6	6	4	5	3	7	5	3	2	5	7	5
	19	問題解決する際、複数の解決案から適切なものを選択できている	5.04	3	2	5	4	5	6	5	6	7	7	4	6	7	4	3	3	5	5	6	4	4	4	7	7	7
	20	チーム内で意見が分かれた場合、それぞれの主張をお互い理解してから解決するようにしている	5.24	4	5	5	2	6	7	5	5	6	5	6	6	7	6	6	3	5	5	5	4	4	4	7	7	6
態度	21	特定の人しか知らないことがないように、ソースコードや業務に関する知識を共有している	4.56	3	5	4	5	5	5	4	6	1	5	6	7	7	3	6	3	5	4	6	5	4	4	1	5	5
	22	チームのメンバは、身勝手な判断や行動ではなく、各自の判断で自律的に動いている	5.00	3	5	5	5	7	6	4	4	7	7	4	7	6	5	5	3	5	4	3	5	5	2	7	6	5
	23	チームメンバ全員が一つの目標に向かった振る舞いを取っている	5.36	6	5	5	5	7	6	6	6	7	5	6	7	7	5	5	3	5	5	4	5	6	4	1	7	6
	24	絶えず顧客と相談することで無駄な機能を作らないようにしている	4.56	5	4	5	6	6	7	6	4	1	4	6	5	6	6	5	4	5	4	5	3	4	4	1	3	5
心身の健康	25	メンバの行動や成果物に対して、メンバが相互にフィードバックを適切に実施している	4.64	2	4	5	5	5	6	4	6	1	5	4	6	5	6	5	3	5	3	6	4	4	2	7	7	6
	26	メンバは、自分や他のメンバが行なっている仕事の目的や重要性を理解している	5.44	5	5	6	4	7	6	5	5	5	6	4	6	6	6	6	4	5	5	7	5	5	4	7	7	5
	27	メンバは有意義な作業のみを実施し、退屈で繰り返し実施する作業はしなくても良い	4.24	1	6	4	6	6	6	7	6	3	5	2	6	5	2	3	2	2	3	6	5	4	3	1	7	5
	28	無理してあとが残かないやり方でなく、健康的でメリハリのある活動を心がけている。また、メンバーの一人一人が安心して自分らしく働いている	4.92	4	6	6	4	7	4	6	6	1	5	6	6	6	4	5	2	4	4	6	5	4	4	7	7	4
		合計		122	140	125	124	163	138	143	158	108	145	122	164	173	131	139	102	123	121	160	133	115	110	112	187	152

付録3-2 アンケート結果（実施しているアジャイル・プラクティス）

			合計	AG_A	AG_B	AG_C	AG_D	AG_E	AG_F	AG_G	AG_H	AG_I	AG_J	AG_K	AG_L	AG_M	WF_A	WF_B	WF_C	WF_D	WF_E	WF_F	WF_G	WF_H	WF_I	WF_J	HY_A	HY_B
予測	1	リリース計画ミーティング	21	●		●	●	●	●	●	●	●	●		●	●	●	●	●	●	●		●	●		●	●	●
		プロダクトバックログ(優先順位付け)	15	●	●	●	●	●	●	●	●		●		●	●							●	●			●	●
		プロダクトオーナー	6						●		●		●		●									●			●	
		人材のローテーション	8				●			●			●							●			●	●			●	●
	2	イテレーション計画ミーティング	9			●		●	●		●	●	●		●	●								●				
		イテレーション	11	●		●		●	●		●	●	●		●	●								●			●	
		日次ミーティング	21	●	●		●	●		●	●	●	●		●	●	●	●	●	●	●	●	●	●		●	●	●
	3	プランニングポーカー	2										●			●												
		イテレーション計画ミーティング	9	●		●		●	●			●	●		●	●								●				
		イテレーション	12		●	●		●	●		●	●	●		●	●								●			●	●
		ベロシティ計測	6	●									●			●							●				●	●
監視	4	かんばん	9	●					●	●	●	●			●								●				●	●
		タスクボード	10	●	●		●			●	●		●		●	●											●	●
		スプリントバックログ	11	●		●		●	●			●	●		●	●								●			●	●
	5	ニコニコカレンダー	2										●			●												
		心理的安全	5						●										●			●	●					●
	6	日次ミーティング	20	●	●		●	●		●	●	●	●		●	●	●	●	●	●	●	●	●			●	●	●
対処	7	ペアプログラミング	5										●			●			●								●	●
		コーディング規約	12					●			●				●	●	●		●	●		●	●	●			●	●
		人材のローテーション	9				●			●		●	●							●	●		●				●	●
	8	柔軟なプロセス	3							●																	●	●
		集団によるオーナーシップ	7						●	●	●				●	●				●							●	
	9	紙・手書きツール	9			●	●	●			●				●	●				●			●				●	
		チーム全体が一つに	11	●		●	●	●	●	●	●	●			●	●											●	
		共通の部屋	10				●	●		●	●				●	●			●		●		●				●	
	10	プロダクトオーナー	7				●		●		●		●		●												●	●
		プロダクトバックログ(優先順位付け)	13	●		●	●	●	●	●	●		●		●	●							●				●	●
学習	11	ペアプログラミング	3				●									●											●	
		スプリントレビュー	6	●			●			●	●				●												●	
	12	ふりかえり	14	●	●			●			●		●	●	●	●			●				●	●	●		●	●
	13	アジャイルコーチ	2	●			●																					
	14	人材のローテーション	7				●			●		●								●			●	●			●	
技術	15	人材のローテーション	5				●			●										●			●				●	
		チーム全体が一つに	9			●	●	●		●	●				●	●											●	●
		ペアプログラミング	5				●									●			●								●	●
	16	人材のローテーション	4				●			●													●				●	
		リファクタリング	6			●		●					●			●							●				●	
	17	プロダクトバックログ	5			●				●						●											●	●
		継続的インテグレーション	9			●				●	●				●	●				●			●				●	●
非技術	18	紙・手書きツール	10			●	●	●			●				●	●				●			●				●	●
		共通の部屋	10				●	●		●	●				●	●			●		●		●				●	
		かんばん	5							●	●				●								●				●	
	19	柔軟なプロセス	8				●		●	●	●	●			●					●							●	
	20	チーム全体が一つに	9			●		●	●	●	●				●	●											●	●
		ファシリテータ（スクラムマスター）	9	●		●		●	●	●			●		●	●				●								
態度	21	集団によるオーナーシップ	8				●		●	●	●				●	●				●							●	
	22	自己組織化チーム	8			●	●	●	●						●	●				●							●	
	23	チーム全体が一つに	8			●		●	●	●	●				●	●											●	
	24	オンサイト顧客	5			●	●		●						●					●								
心身の健康	25	迅速なフィードバック	8			●	●		●		●		●		●											●	●	
	26	インセプションデッキ	2						●				●															
		集団によるオーナーシップ	7						●	●	●				●	●				●								
		自己組織化チーム	8			●	●	●	●						●	●				●								
	27	自動化された回帰テスト	11				●	●	●	●	●				●	●				●			●				●	●
		継続的インテグレーション	10			●	●	●		●	●				●	●				●			●				●	
	28	ニコニコカレンダー	2										●			●												
		持続可能なペース	8			●	●	●		●	●				●	●											●	
			474	16	6	24	31	27	26	31	33	13	25	1	39	40	4	3	10	21	6	4	24	13	1	4	47	25

付録3-3 アンケート結果（コメント）

No	質問	AG_A	AG_B	AG_C	AG_D
1	プロジェクト計画時、顧客やメンバと話し合い優先順位とリリース時期を明確にした上で計画を作成している 対人的・時間的リソースにリスクは無いか？特定の人しかできない業務はないか？といったことまで考慮している	・開発ロードマップ、開発要件（開発STEP表）、顧客からの ヒアリング情報をインプットに、開発計画を策定。 ・緊急度が高く、短期開発案件はアジャイル開発プロセスを適用し、小チームで実施している。 ・短期開発を実現するため、予めスキル開発要件見合うメンバで構成している。			
2	プロジェクト中にリスクが発覚した場合、具体的な内容をチームメンバ全員が共有・理解した上で作業を進めている 例： 実現性や不明点のリスクなどを事前に把握し、チームで共有している	・メンバ内で日次ミーティングで共有。 （共有はするが、全員の理解を求めていない） ・プロジェクト全体のリスクはリーダMTG（週次の上位会議）にて解決している			メンバ内および対顧客でそれぞれ日次ミーティングを実施している。
3	要件変更があった場合、計画変更を行いチームとして対応が可能な範囲を再定義している 例： 人的・時間的リソースや負荷状況をチームで共有した上での、実現性のある計画を立てている	要件変更の影響を受けるメンバを対象に都度計画。 ベロシティ計測が参考になれば活用。			要件変更がある場合は顧客とミーティングを行い、再度優先順位付けを実施して、作業内容を再定義している。
4	メンバの負荷状況やプロジェクトの進捗状況が見える化できており、異常な状態であると判断するメトリクスを設定している	・１日にできる作業量を確認している（逸脱は翌日調整） ・メトリクスなど閾値を設けているわけではない。 ・かんばんツールは『TRICHORD』を利用		Redmine，JIRA使用。	Redmineによる進捗管理により、メンバ毎の負荷状況を把握している。
5	メンバが報告しやすい状況・環境が整っている	・若手メンバを集めて開発チームを構成。 ・互いに気になることを日々コミュケーションをとっていた。	デイリースクラム、レトロスペクティブで誰でも発言できる雰囲気を意識して作っている		
6	深刻な問題を招く恐れのある状況が見落とされることなく、問題として認識できる	・日次ミーティングにプロダクトリーダも加わり状況を確認。 トピックをプロダクトオーナへの報告、必要に応じて協議する場がある。			
7	メンバーが急に減った(例：入院した)場合、そのメンバーの担当していた作業を短期間で引き継げる	・（１ヵ月で完成させるなど）短期開発でのみ適用している。 これまで、メンバが減るなどの状態は発生していない。 ・随時レビューを実施しているので引継ぎなくフォローは可能	チーム内で決めている最低限のモデル（クラス図、シーケンス図、状態遷移図）はドキュメントとして残している		メンバ内でのリカバー可能なように意識的なジョブローテーションを行うとともに作業平準化のための手順書を作成している。
8	問題の解決のためにメンバ間の作業調整を行う際、特定のメンバに作業が集中しないように分担できている	・日次ミーティングで調整。 ・プロダクトとしてはクリティカルパスのみチェック			
9	解決が困難な問題が発生した場合、問題の解決のためにメンバー同士が協力して対応できる	・日次ミーティング後、随時打合せ ・専用で利用できる開発ルームがあり、必要に応じて集中検討会を開催		Microsoft Teams，Redmine使用。	
10	影響の大きな設計変更や機能追加が必要となった場合、柔軟に計画見直しやプロセスの見直しができる	・開発対象を絞って適用しているため、大きな変更が入らないように進めている。 ・小規模な変更、追加はあるがプロダクトバックログを活用			
11	スキルの高いメンバーが、低いメンバーに対して、日常的に指導している	・スプリトレビューは実施しているが教育目的には使用していない ・計画段階でプロジェクトチームの技術力を把握し構成している。			
12	開発途中で、今までの作業の振り返りを行い、良い結果をチーム内で共有し、他のメンバも実践している				
13	新たな開発手法を取り入れた時、その開発手法の有識者や経験者に直接指導を受けている	・初めてアジャイル開発を回す場合は、（社内の）アジャイル開発コーチに参画してもらい、都度助言をもらっている。			最初はM2Mで指導を行っている。
14	会社やチームがメンバの学習する環境を積極的に提供している	・社内にアジャイル開発入門講座が提供されている。 アジャイルプロセス開発のメンバは事前に講座を受講させて いる			関連する技術に対する勉強会などがあれば積極的に参加している。
15	開発メンバが保有する固有の技術スキルについて、メンバ間での共有(学習・教育含む)が適宜行われている	・先端技術、固有技術は（"QU活動"という）チームでの改善活動のフレームワークに載せて、メンバ間で共有を図っている。 ・基礎技術は社内講座を活用。	手順書を作成し展開している		
16	技術スキルが不足しているメンバでも作業が滞りなく実施できるよう、開発作業に必要となる情報はドキュメント化がされている状態である	・スキルを補う手段としてのドキュメント化は実施していない ・定型的な作業は手順書を随時作成している			
17	将来の開発において必要となりうる技術スキルに関し、チームとして習得に向けた活動(勉強会など)を作業として計画している	No15と同じ。			
18	メンバ間での情報共有が活発に行えている	・非技術スキルについてもNo15の活動を展開している。 ・計画的に実施しているものであり、"活発"に該当するものではない。		Microsoft Teams，Redmine使用。	個人ごとにA4サイズのホワイトボードを利用して、図やシーケンスなどを使用した説明を行っている。
19	問題解決する際、複数の解決案から適切なものを選択できている	■開発段階で発生した問題については ・主導するメンバの見解に対しメンバ間で協議している （必ず対案を出すというようなことはしていない） ■調査検討段階でのまとめの場合は ・必ず複数案を提示し、メリット・デメリットを表にしてもまとめ、結論を出している			
20	チーム内で意見が分かれた場合、それぞれの主張をお互い理解してから解決するようにしている	No.19のやりかた。 ・まとまらない場合は主導メンバがファシリテータとなって意見を集約し解決案をまとめている			
21	特定の人しか知らないことがないように、ソースコードや業務に関する知識を共有している	・レビューは随時実施している。			
22	チームのメンバは、身勝手な判断や行動ではなく、各自の判断で自律的に動いている	・タスクにして、計画的に作業している。 ・コントロールはできているが自律性は考慮していない			
23	チームメンバ全員が一つの目標に向かった振る舞いを取っている	・短期で開発するものをターゲットとして適用しているため、目標（ゴール）意識は高い			
24	絶えず顧客と相談することで無駄な機能を作らないようにしている	・スプリトレビューに加え、週次で進捗・状況報告を行っている。			日次ミーティングを実施している。
25	メンバの行動や成果物に対して、メンバが相互にフィードバックを適切に実施している	・相互にフィードバックするケースは少ない。 ・日次ミーティングにて互いの状況は把握している。			成果物を共有するとともに、事前レビューを実施している。
26	メンバは、自分や他のメンバが行なっている仕事の目的や重要性を理解している	・開発の方向性、目標（ゴール）は理解している。 ・特別なテーマのみ適用しているプロセスなのでメンバの関心や意識も高い。			
27	メンバは有意義な作業のみを実施し、退屈で繰り返し実施する作業はしなくても良い	（質問の意味が捉えられない） ・繰り返し作業もタスクには存在する。 ・有意義/退屈と考えるかは本人次第。			自動化されたリグレッション試験を実施している。
28	無理してあとが続かないやり方でなく、健康的でメリハリのある活動を心がけている。また、メンバーの一人一人が安心して自分らしく働いている	（質問の意味が捉えられない） ・短期集中開発を実現させるためのプロセスとして採用しているので、多少無理は発生する。 ・"安心して"、"自分らしく"と思うかは本人次第。 （プロジェクトの枠組みではメンタルケアは実施していない。労務管理として別枠でケアしている）			

No	AG_E	AG_F	AG_G	AG_H	AG_I	AG_J	AG_K
1	少数精鋭で柔軟かつスピーディーに開発を推進しており、多少属人的になってしまうリスクが潜んでいる。				優先順位とリリース時期は明確にするが、顧客ではなく営業は販売会社になる。また、特定の人の業務はないか、などは考えていない。	優先度はつけていて、リソースにリスクが無いかは確認しつつ、なかなか厳しい状況が続いている。しかし、どのくらい厳しいのかは、ポイントをつけることでわかっている。	顧客とは話していない
2		Slackおよびかんばんでの情報共有			毎週の定例会で共有。毎日ではない。		
3							
4	毎日の朝ミーティングでバックログ、かんばんの状況を踏まえた意識合わせを行っており、プロジェクトのヘルシー状態を把握している				作業時間をメンバーに入力して見える化としている。メトリクスや明確な判断基準はない。	チームメンバがそれぞれ違うPJを受けもっていることが要因と思われるが、タスクボードを見るだけではなかなか負荷状況が見えてこない。	ニコニコカレンダー
5	コミュニケーションツール（Slack、Mattermost）を導入し、ロケーションや時間に縛られない開発環境を構築。 ※海外出張中も連携可能				定例会ぐらいしかない。	(アンケートへのコメント。心理的安全って、スローガンのように捉えているのですが、具体的なプラクティスになってるんでしょうか。)	
6	毎日の朝ミーティングにおいて、各メンバから課題情報の収集を行っております。	Slackでの情報共有			見落としは少なくない。		
7	多少属人化はしているが、共通ライブラリやフレームワーク技術を取り入れるなどして、個々の個性を最小限としている。また、少数精鋭で個々のスキルが高いため、他人の分野でも少し時間をかければメンテナンスすることができる。	コードレビュー			短期間で引き継げていないが、やらない選択肢はないので、今いるメンバーで何とかする考えになる。	仕事としてバディ制にしているので、急に減った場合にも対処しやすい。また、そのため、プライベートの用事も優先しやすい状況にある。	
8	スピードを優先していることから、得意分野に応じて、バックログをアサインするようにしていることから、多少、特定のメンバーに作業が集中してしまうこともある。（※若手エンジニアは教育対象のため上級エンジニアに付いて幅広くアサイン）				分散しようとしているが、現実は偏っている。		
9					複数のプロジェクトを並行しているので、同プロジェクト間は協力し合えているが、他プロジェクトまではできていない。余裕がない。		共通の部屋でなかったが専用の作業スペースを確保
10					できる日程で調整する。		
11	ペアプログラミングとまでは言えないが、若手エンジニアが上級エンジニアに付いて、レビューを受けながら開発を進めるようにしている	コードレビュー			日常的にはしていない。		ペア設計
12					開発途中の振り返りはやっていない。		
13	文献や外部（NTTデータ主催）のアジャイルセミナー等に参加し、開発の進め方について勉強している				指導まではしない。	課全員で教育を受ける	
14	Udemyの活用 展示会やカンファレンスの参加 ※上記の活動を後押し				提供はしている。		
15	定期的に勉強会を開催。 ※新しい技術の紹介等	コードレビュー	内部勉強会		やろうとするが、現実は忙しくてできていない。	互いの成果物やノウハウを定期的に共有する場を設けている	
16	少数精鋭で進めることを前提としており、ある程度のスキルレベルが必要となっている				アップデートされていないドキュメントが多い。		
17	先端技術は積極的に試し、勉強会で共有しあっている				計画はされていないし、チームとして取り組もうとしているが現実は忙しくてできていない。	タスクボードに挙げているが、なかなか実践できていない	この設問とCIは関係ないのでは？
18		Slackでの情報共有	Viber, Teams等SNSを用いたコミュニケーション				
19	チーム全体で各案の利点欠点の比較検討を行い、解決策を選択している					KPT（複数のTryを挙げて解決策を検討）	
20					ほぼできている		
21		コードレビュー			やろうとするが、現実は忙しくてできていない。		
22	少数精鋭で信頼関係を構築して進めている。 開発本来の目的に影響するような部分は、共通の価値観で判断し、チームに相談するようにしている。						
23							
24	こまめにデモを行い、方向性を合わせている				営業や販売会社と直接相談することはほとんどない。	(開発ではないので、設問が対象外かも)	顧客ではなくシステム担当者、ハード担当者と絶えず相談している
25					フィードバックはない。	スプリントレビューのタイミングがメイン。	
26	キックオフ資料という形でインセプションデッキを作成し、こまめにアップデートしている				自分の業務の重要性は理解しているが、他のメンバの業務の重要性は理解できていない。	タイムラインによる長期のふりかえり	
27	単体テストおよびリグレッション試験（UI結合テスト）を自動化する仕組みを構築。 Jenkins、Github、Selenium				繰り返しの作業も必要ならやる。やらない選択肢はない。		
28	すべに自動テスト、CI化できたわけではなく、 5年間で作り上げたチームである				残業時間が多い。多分不健康。		

No	AG_L	AG_M	WF_A	WF_B	WF_C	WF_D	WF_E	WF_F	WF_G	WF_H	WF_I	WF_J	HY_A	HY_B
1	WFでは特定の人にしかできない業務があること自体、課題になるという思想がそもそもないので 点数が開いた	顧客とは話していない			プロダクトバックログのようなものとして、要件の優先順位をまとめた資料は存在する。			プロジェクト計画書レビュー						
2	WFではチームメンバー全員が共有するという思想もないため点数が開いた							朝会や週次の定例会でリスクの共有を行っている						
3	変更対応はWFでもアジャイルでも同じように発生する				要件変更時の計画変更は、顧客と話し合いながら実施できている。				チケット駆動開発			やるしか選択肢はない。		
4	WFでは、メンバーの負荷状況は、見える化することになってないのでわからない	ニコニコカレンダー			Redmineを使用している	Redmine,Excelでの日程計画		RedmineまたはExcelによる進捗管理				個人管理＆非公開		
5	報告しやすいか？どうかは、リーダ次第のところがあるかも							・メンバー同士が物理的に距離が近い ・コミュニケーションツールの使用				Daily段コミ	1 on 1ミーティング	
6	朝会をしなくても、リーダが個別に担当者のところに行けば状況はわかるが、WFでは担当者同士で気づく問題の検出ができない													
7	WFの場合、引き継ぎやすい開発をする思想はない				コーディング規約は存在するが、守られていない状態。							代わりはいない。		
8	WFの場合、特定のメンバーが集中しない開発をしないとする思想はない								チケットベースの作業調整			一部に集中。		
9	WFの場合、協力し合うという思想はそもそもない	共通の部屋でなかったが専用の作業スペースを確保										Teamsでチーム内連携を取っている。		
10	WFは、はじめに決まった要件が前提で最後までやりきるやり方なので、開発中の機能変更が発生することは、アジャイルより柔軟にできるやり方ではない				No.3と同様							顧客要求は必達なので、自由度は少ない。		
11	WFは、知識を共有。知らない人に素早く広めたいという意識が、明確になっていない	ペア設計							仕様書レビュー・コードレビュー			暇はない。		
12	WFの振り返りは周期が数か月と長いため、カイゼンサイクルが長いが、スクラムの場合、周期が２週間なのでカイゼンサイクルを回しやすい				ふりかえりをしても、あまり成果が出ていない							暇はない。		
13	アジャイルの嬉しさや、良いやり方であることが、チームとして共有できているとは言えない やることになったのでやってます みたいなところがある				社内有識者に話を聞いている。							取り入れない。		
14	少しずつはやっているが、ローテーションの方針が明確になっていない				勉強会や展示会は行ける雰囲気。							課長の方針はあるが、暇はないので誰も行動しない。		
15	違いを感じない											暇はない。		
16	現状のアジャイルチームの方はやっていると感じる											一部の人が実施。		
17	現状のアジャイルチームの方はやっていると感じる	この設問とCIは関係ないのでは？							勉強会			暇はない。		
18	現状のアジャイルチームの方はやっていると感じる					redmine						日次ミーティングとTeams		
19	違いを感じない											ブレストしている。		
20	現状のアジャイルチームの方がやっていると感じる													
21	現状のアジャイルチームの方がやっていると感じる					SVN			ドキュメント化			暇はない。		
22	現状のアジャイルチームの方がやっていると感じる													
23	現状のアジャイルチームの方がやっていると感じる											個々人は別の目標。		
24	現状のアジャイルチームの方がやっていると感じる	顧客ではなくシステム担当者、ハード担当者と絶えず相談している							事前に販売会社・企画部門と合意した要求仕様に基づく開発			要求は全て必要。		
25														
26														
27												手動のみ。		
28									KI活動			残業規制により、課長以外は早く帰宅している。		

付録3-4 アンケート結果（レジリエンスに対する自己評価とプラクティスの相関係数）

凡例 >=0.7 >=0.4 その他

能力		アジャイル・プラクティス	予測			監視			対処				学習				技術			非技術			態度				心身の健康			
			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
予測	1	リリース計画ミーティング	0.59																											
		プロダクトバックログ(優先順位付け)	0.58																											
		プロダクトオーナー	0.27																											
		人材のローテーション	0.36																											
	2	イテレーション計画ミーティング		0.40																										
		イテレーション		0.48																										
		日次ミーティング		0.69																										
	3	プランニングポーカー			0.65																									
		イテレーション計画ミーティング			0.40																									
		イテレーション			0.38																									
		ベロシティ計測			0.48																									
監視	4	かんばん				0.54																								
		タスクボード				0.46																								
		スプリントバックログ				0.20																								
	5	ニコニコカレンダー					0.43																							
		心理的安全					-0.03																							
	6	日次ミーティング						0.46																						
対処	7	ペアプログラミング							0.55																					
		コーディング規約							0.37																					
		人材のローテーション							0.32																					
	8	柔軟なプロセス								0.86																				
		集団によるオーナーシップ								0.69																				
	9	紙・手書きツール									0.48																			
		チーム全体が一つに									0.44																			
		共通の部屋									0.33																			
	10	プロダクトオーナー										0.52																		
		プロダクトバックログ(優先順位付け)										0.42																		
学習	11	ペアプログラミング											0.75																	
		スプリントレビュー											0.33																	
	12	ふりかえり												0.89																
	13	アジャイルコーチ													0.50															
	14	人材のローテーション														0.38														
技術	15	人材のローテーション															0.56													
		チーム全体が一つに															0.61													
		ペアプログラミング															0.24													
	16	人材のローテーション																0.25												
		リファクタリング																0.38												
	17	プロダクトバックログ																	0.63											
		継続的インテグレーション																	0.62											
非技術	18	紙・手書きツール																		0.57										
		共通の部屋																		0.43										
		かんばん																		0.53										
	19	柔軟なプロセス																			0.44									
	20	チーム全体が一つに																				0.64								
		ファシリテータ（スクラムマスター）																				0.25								
態度	21	集団によるオーナーシップ																					0.61							
	22	自己組織化チーム																						0.62						
	23	チーム全体が一つに																							0.79					
	24	オンサイト顧客																								0.61				
心身の健康	25	迅速なフィードバック																									0.81			
	26	インセプションデッキ																										0.37		
		集団によるオーナーシップ																										0.28		
		自己組織化チーム																										0.41		
	27	自動化された回帰テスト																											0.76	
		継続的インテグレーション																											0.66	
	28	ニコニコカレンダー																												0.20
		持続可能なペース																												0.70

付録4　将来，レジリエンス能力を改善をする方法が確立した場合の提案

現状、アンケートなど改善課題が多いが、将来、現場で使えるようになった場合の
レジリエンスの改善方法をここに記載する.

　方針：自チームの弱点を分析し改善する

＜すすめ方＞
STEP1：自チームのアンケート調査結果からレジリエンス毎にレーダーチャートを作成する
STEP2：レジリエンス能力の高いチームのデータと自チームのデータを比較する
STEP3:レーダーチャートから、弱点を分析し改善をすすめる.
レジリエンス能力の高いチーム程、レーダーチャートの面積は広くなる.

今回は、例としてアンケートの平均点が低いAチームと高いBチームを選択して
以下の様に比較した.

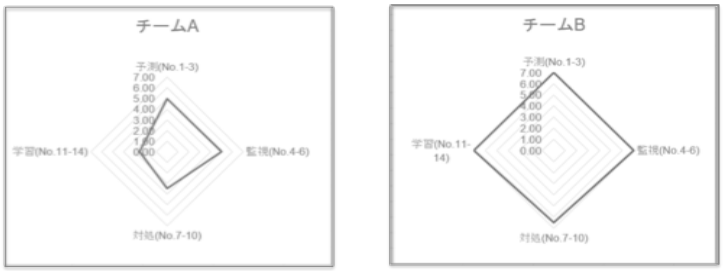

図1　　基本能力の比較

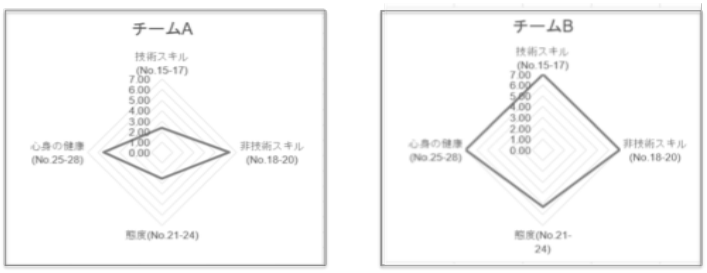

図2 レジリエンスを支える要素の比較

図1より、学習」に対し Aチーム：2.5, Bチーム：7.0と大きな違い(4.5)があった.
図2より、技術スキルに対しAチーム：2.3, Bチーム：7.0と大きな違い(4.5)があった.

点数の低かったAチームでは，いずれも「暇がない」とコメント欄にあったが
アンケートの内容（付録２アンケート調査票No12,17を抜粋）は、それぞれ、
「開発途中で,今までの作業の振り返りを行い,良い結果をチーム内で共有し,他のメンバも実践している」,「将来の開発において必要になりうる技術スキルに関し,チームとして習得に向けた活動(勉強会など)を作業として計画している」など
大きな違いがあった点数のアンケート内容を参考にすることで、レジリエンスの改善へつなげることができるのではないか.と考える.

付録5　アンケート高得点チームのプラクティス分析

アンケートの合計点が高得点であった上位２チームチームが共通して実践しているアジャイル・プラクティスを調べたところ，「プランニングポーカー」，「リファクタリング」，「持続可能なペース」であった．このプラクティスの効果を以下に記載する. レジリエンスが低いチームが今後，どのようなアジャイル・プラクティスを選択するとよいかを考える上でヒントになると考える.

プラクティス	効果
プランニングポーカー	メンバの業務経験や知識を生かすことができる
	ゲーム感覚で楽しく活発な意見が引き出せる
リファクタリング	わかりやすくつくり，再利用しやすくすることで、将来へのリスクを軽減する活動になる
持続可能なペース	高負荷やストレスが高い状況が続く状況を放置しない
	ゆとりと活気をもって効率的に仕事ができるチームをめざす