令和 2 年度補正経済産業省デジタルプラットフォーム構築事業（補助金申請システムの機能開発等に係る技術的支援を通じたマイクロサービス化等に関する調査実証事業）

調査報告書

1. 本事業の概要

経済産業省は、従来の補助金申請電子化の課題を解決し事業者の利便性・生産性の向上に資するため、「平成30年度補正経済産業省デジタルプラットフォーム構築事業（補助金申請システムの構築等事業）」等を実施し、令和2年1月に汎用的な補助金ネット申請サービスである jGrants1.0 の運用を開始した。

jGrants1.0 は、経済産業省の扱う補助金のみならず、他省庁・自治体の補助金を含めて複数の補助金をワンストップでチェック、選択可能なほか、公募から交付・事業報告までプロセス全体を電子で完結することが可能である。また、補助金手続に必要となるデータ項目の簡素化・標準化や、部分的ではあるが事業者の申請入力に対するワンスオンリー機能を提供することで、事業者の入力負担の軽減を実現している。一方で、「標準的な補助金手続フローを念頭に機能を実装したため、より複雑な申請プロセスを設定している補助金では利用が困難である。」「より多くの事業者の利用を促進するためには、ワンスオンリー対象項目の拡充等による利便性の向上や、電子申請に不慣れな利用者に対するヘルプデスク等の支援の拡充が必要である。」といった課題がある。

経済産業省は、これらの課題解決や、利用者から寄せられた要望に応えるために「令和2年度補正経済産業省デジタルプラットフォーム構築事業（補助金申請システムの機能開発等事業）」（以下、jGrants2.0 開発事業という。）等において後継システムである jGrants2.0 の機能開発を行うこととした。

jGrants2.0 は、サービス提供開始後も利用機関の拡大や利用補助金の多様化を念頭に、継続的に機能追加や改善が行われることが想定される。したがって、サービス運用主体には各サービスや稼働環境の整合を確保し、サービスの稼働やセキュリティを担保するためのプロジェクトルールの確立が必要となる。

本事業は、このプロジェクトルールの確立のための知見の収集、整理を行い、jGrants2.0 開発事業における設計・機能開発等の支援を通じて、これらのルール等の実証を行うものである。

2. 実証したプロジェクトルール

2.1. 開発スタイル

Mini-WF と Agile のハイブリッド型

2.1.1. フェーズ別に異なる開発モデル with V-Shape Model

- 基本的に、1 つの PRD を作成する度に 1 回のウォーターフォール(WF)と 1-n 回の開発スプリント(Agile/Scrum)を回す、モデルハイブリッド型の開発スタイルとする。
- ドキュメントフェーズでは、why(なぜ作るのか), what(何を作るのか), How(どのように作るのか)毎に、MRD, PRD, DevSpec,TestSpec に分けてドキュメントを作成（情報の疎結合化）し、レビュー合意（サインオフ）することが主なアウトプットとなる。
- MRD > PRD >DevSpec の順に情報が引き継がれていくが、TestSpec の作成に関しては PRD のサインオフ直後から開始することができ、作成途中に DevSpec の情報を付与して完了とすることができる。
- 開発実装フェーズでは、各 PRD で明記した要件を基にスプリントを回し、最低限デプロイできる状態まで実装を完了させることが主なアウトプットとなる。
- 実装後のシステムテスト(ST/QA)、ユーザー受入テスト(UAT)は複数 PRD の開発実装アウトプットをまとめて実施可（上図の IT まで完了したら、全体テストまでは止めておくことができる）。
- ドキュメントフェーズの各ドキュメントと対応したテストの関係を上図の点線矢印で示しており、各テストの粒度設定やバグトリアージの優先度付けに活用する。

2.1.2. 複数の Mini-WF と Sprint

- 従来の WF は、1つの大きな要求フェーズ、1つの大きな要件フェーズ、1つの大きな設計フェーズを順に fix させて進めていたが、本開発スタイルでは各フェーズを課題単位、要件機能単位、実装機能単位毎にドキュメントを分けることで、複数の小さな流れ（Mini-WF）を同時並行的に進めていくことができる（ドキュメントのマイクロサービス化）。
- 各ドキュメントがサインオフされ、実装するための優先度を決定した後に、Scrum/Agile の開発手法に落としこみ実装フェーズをスプリントで回していく。
- 実装完了した機能は Git などで管理・蓄積されていき、ST/QA（総合テスト、システムテスト、QA）ができる状態となったらテストを実施する(WF の全体テストに相当)。

2.2. ツール

用途	ツール
コミュニケーション	Slack
文書作成・管理	Confluence
チケット管理	Jira

2.3. ドキュメントプロセス（サインオフプロセス）

情報単位にドキュメント化し合意形成をとる WaterFall プロセス

2.3.1. 役割

Role	
PO	プロダクトオーナー
PM	プロダクトマネージャー
DEV	開発チーム
QA	テストチーム

2.3.2. ドキュメント

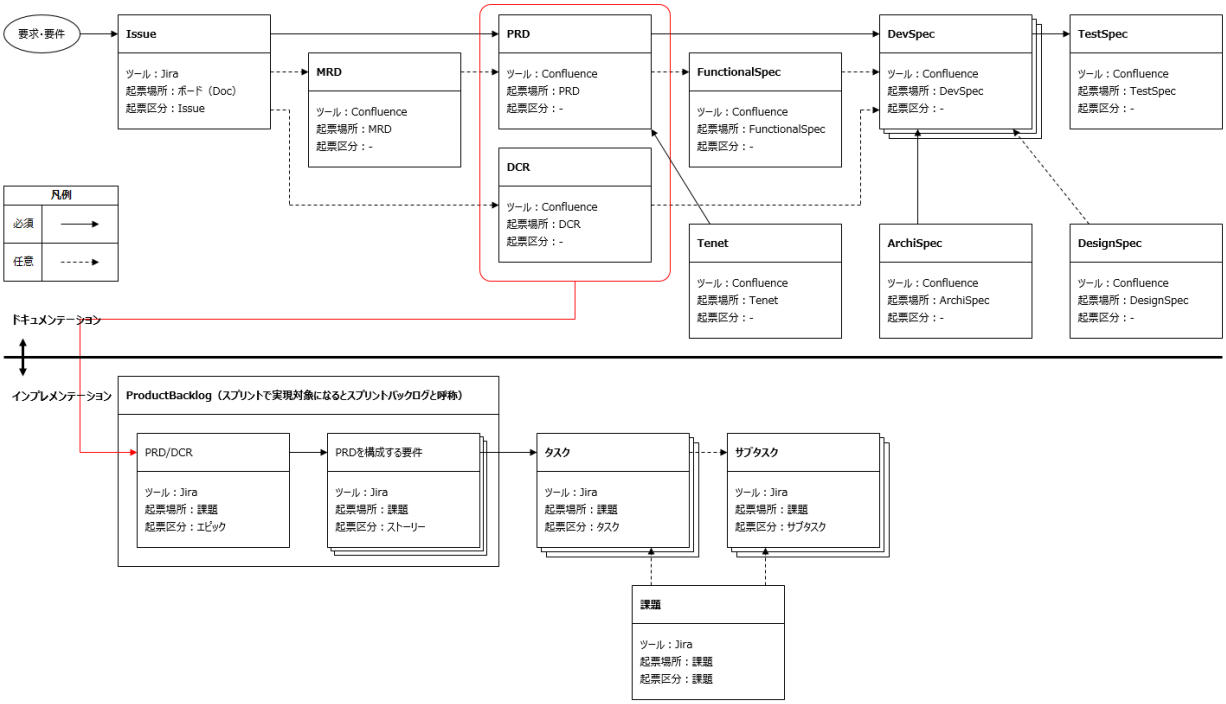
必須ドキュメント

名称		説明
PRD	Product Requirements Document 製品要求仕様書	• 何を作るかを明記した構造化フォーマット • スコープや要件などをわかりやすく記載 • 基本的に技術的な解決方法については記載しない（技術上の可能性を制限しないため）
DevSpec	開発仕様書	• どう作るかを明記した構造化フォーマット • アーキテクチャや利用技術、要件との紐付けなどをわかりやすく記載
TestSpec	テスト仕様書	• どう品質を担保するかを明記した構造化フォーマット • テストの要件やスコープ、計画やリスク、テスト実現性、テスト手法などを記載 • テストケース/テストシナリオは別で作成し、TestSpec の補助ドキュメントとする

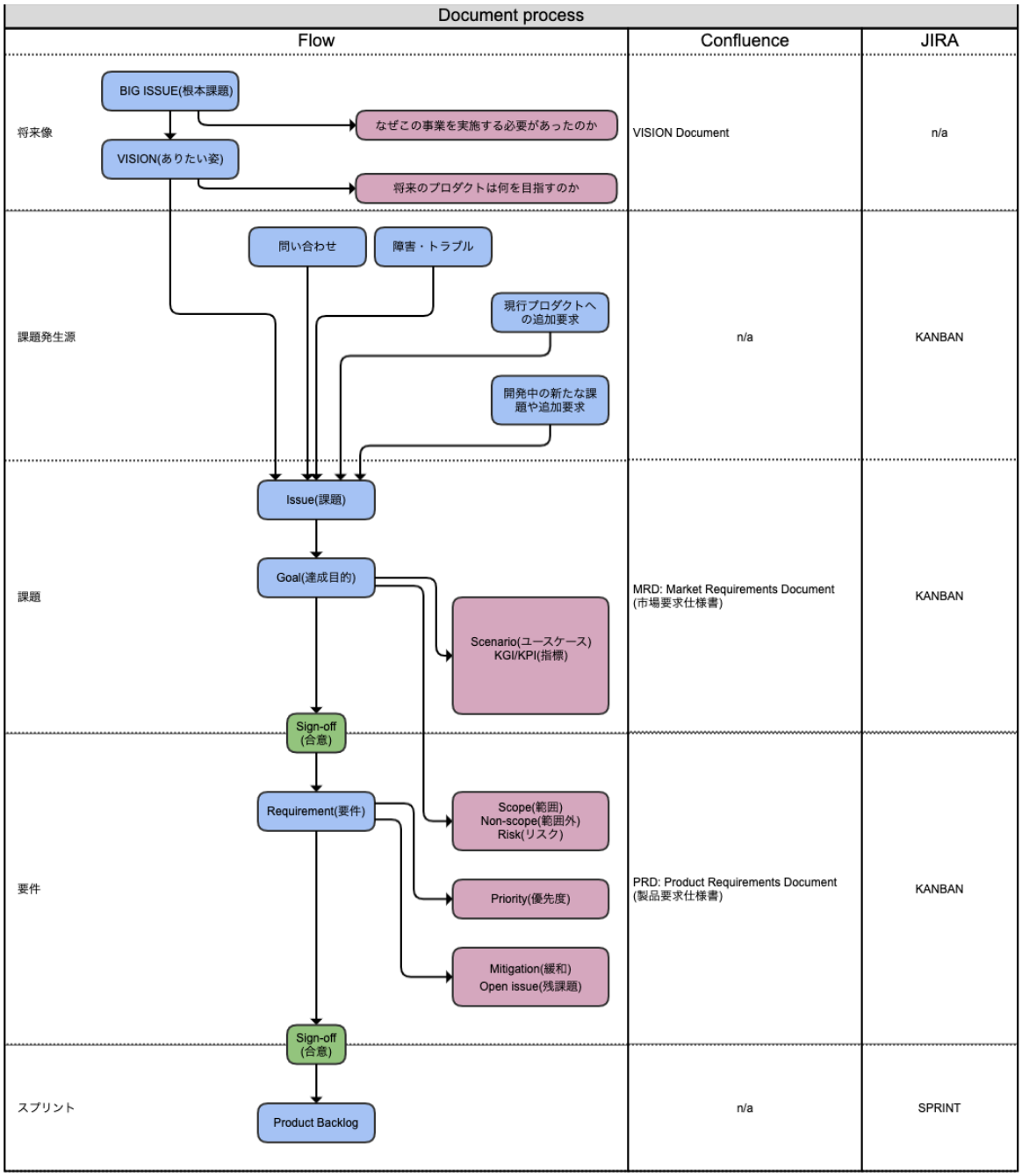
任意ドキュメント

名称		説明
MRD	Market Requirements Document 市場要求仕様書	• なぜ作るかを明記した構造化フォーマット • 課題やゴール、ユースケースなどをわかりやすく記載 • MRD を作成しない場合は PRD に内包し、課題やゴール、KGI/KPI、シナリオを PRD の冒頭に記載する
DCR	Document Change Requirement	• PRD の簡易版 • DCR は PRD ほどの要件規模ではなく、UI の一部を修正したりするような場合に記載項目やプロセスを簡略化できる
FunctionalSpec	機能仕様書	• 何を作るかを具体的な機能観点で明記した構造化フォーマット • 外部 API の I/F など、ユーザー（例えば外部システムのこと）に直接影響する機能要件を記載する
DesignSpec	デザイン仕様書	• UI/UX デザインについて明記した構造化フォーマット • 画面フレームやカラー指定、HTML の ID など詳細まで記載 • 必要に応じてモックも記載
ArchiSpec	Architecture Spec アーキテクチャ仕様書	• 全体アーキテクチャについて明記したマスターフォーマット • 詳細機能については記載しない • 必要に応じて各 DevSpec に埋め込む
Release Report	リリースレポート	• リリースした内容について報告書ベースで記載 • リリースで何を達成したかや残課題について記載
Tenet	テネット	• 全ての PRD において共通的に適用すべき要件等 • 共通化できる要件を個々の PRD において記載する負荷を軽減できる

PRD プロセスで作成されるドキュメント相関図



2.3.3. PRD プロセス



- プロダクトが将来どうあるかを Big picture などを用いて VISION ドキュメントとし、メンバー全員に共有する(See a same page)。
- ユーザー問い合わせやシステム障害などのトラブル、追加要求などをチケットベースで蓄積する。
- 開発中に発生する新たな課題や追加要求などをチケットベースで蓄積する。
- Jira 上の課題チケットは優先度が高い順に上から配置する。

- 課題チケットの優先度が高いものから MRD を作成する（チケットとドキュメントの紐付けを行う）。
- MRD は課題提供者（原課やヘルプデスクなど）が基本的に作成するが、ヒアリングを基に PM が作成しても良い（状況や成熟度による）。
- MRD に対してステークホルダーがレビュー後同意する（サインオフ）。
- MRD を基に PRD を作成する（詳細は後述）。
- PRD に対してステークホルダーがレビュー後同意する（サインオフ）。
- サインオフ後、PRD の各要件を Product Backlog のチケットに移す。

2.3.4. PRD

PRD について

- プロダクト要求仕様書(PRD: Product Requirements Document)は、マーケット要求仕様書(MRD: Market Requirements Document)によって明らかとなった課題に対しての解決策（業務要件レベル）をまとめたドキュメントである。
- 基本的に MRD と PRD は分けるべきであるが、解くべき課題がシンプルである場合は MRD はコンパクトにして PRD に内包するのが良い。
- PRD はシンプルに構造化されたフォーマットで、最も重要な情報はリスト化された各要件とその優先度である。
- 大規模なプロダクトでは複数の PRD に分けて記載するのが良い（フィーチャー単位が望ましい）。
- ユーザーに直接影響する情報は PRD に記載する（画面遷移や画面フレームも含む）
- 基本的に PRD には技術的な解決方法を記載しない（技術の選択肢を制限してはいけない）。

PRD のフォーマット

MRD を内包したシンプルな PRD の項目例

項目	記載内容	例
タイトル	• 何を作るのかを簡潔に記載 • タイトルを見ただけで機能がわかるようにする	事業者にとって最適な補助金情報を検索できる機能
ドキュメント	関連するドキュメントのリンクを記載	• 用語集のリンク • 関連する他 PRD のリンク • DevSpec のリンク • TestSpec

用語	本 PRD 内で使用される専門・特殊用語を記載	D2P: Doctor to Patient の略で医療機関と患者間の情報交換のこと(cf. D2D)
課題	• 解決するべき問題が何かを明確に記載（複数可） • 課題設定を間違えると要件が変わるため慎重に落とし込むこと • 本来は MRD に記載される項目	補助金一覧の情報が膨大な量のため、事業者が探している補助金が見つからない
ゴール	• 問題を解決する達成目標を記載（複数可） • ゴールが PRD に書かれる機能の達成目的となり、これを間違えると要件が変わるため慎重に落とし込むこと • 本来は MRD に記載される項目	必要とする補助金を事業者が容易に検索・参照ができること
シナリオ	• ゴールを達成したことにより、誰がどのようなメリットを享受するのかユースケースがわかるように簡潔な文章化して記載（複数可） • より多くのシナリオパターンを出すことで、要件も多く出せることを手助けできる • 本来は MRD に記載される項目	オンライン遠隔医療(D2P)のシステムを開発することとした A 社は、検索画面から IT や医療などのキーワード入力、必要な情報をタグやフィルターで絞りこみ操作をすることで、ニーズを満たす申請可能な IT 補助金を見つけ出すことができた

KGI/KPI	• ゴールが達成できたかどうかを判断できる指標を記載（複数可） • できる限り定量化し、PRD をリリースした後に追えるようにする • 本来は MRD に記載される項目	• 検索後の離脱率(30%未満) • 検索結果数(5 補助金/ページ) • 検索レスポンスタイム(1sec 未満)
スコープ/ノンスコープ	機能の対象範囲と対象範囲外を明確に記載（複数可）	• PC やスマホの全てのブラウザが対象 • ネイティブアプリは対象外
要件/優先順位	ゴールを達成するためにどのようなことが必要か、最低限何が必要かを簡潔に記載（複数可）	見つかった補助金が公募期間前もしくは公募期間中であること(優先度 P1)
リスク	課題を解決することで起こりうる問題や障害を記載（複数可）	検索結果によってはユーザーの離脱や問い合わせを増加させてしまう
ワークアラウンド(緩和策)	要件に対して機能実装以外に暫定対応や運用で回避・緩和できる場合にその対応策を記載（複数可）	検索フォームを用意する代わりに既存のチャットボットで検索を可能とする
詳細	各要件の詳細内容を記載（複数可）	検索日時が 2020/6/1 10:00 の場合、検索結果に表示される補助金の公募終了日時は 2020/6/1 10:00 以降となり、公募開始日時は 2020/9/1 以前となる ※3 ヶ月先までの公募開始日を対象とした場合
リリース依存関係	機能毎にリリースする順序が必要な場合、その関係性を記載する	補助金データ参照 API をリリース後に検索機能をリリースする
オープンイシュー(残課題)	現時点で要件を確定できないため、一時的に保留としておく課題を記載（複数可）	事業者にとって最適な補助金をレコメンドするためのデータを中企庁と調整中

サインオフ	ステークホルダーからのレビューを受けてレビュー済みのサインを記載する ※開発チームとテストチームからのサインは必須	• 開発チームリーダー • 霞ヶ関太郎 • 2020/6/1

2.3.5. 要件の優先度

PRD の各要件(Requirement)に付与される優先度(Priority)の考え方

Priority	Meaning	Remark
P1	本 PRD で設定したゴールを達成するために最低限必要不可欠な要件	P1 が一つでも欠けると PRD は未達成となる
P2	本 PRD で設定したゴールを達成するために最低限必要不可欠ではないが、技術的、スケジュール的に可能な限り満たしておくべき要件	現時点での P2 であり、将来優先度は変更される可能性がある
P3	本 PRD で設定したゴールを達成するために必要不可欠ではなく、現時点では対応しなくても良い要件	現時点での P3 であり、将来優先度は変更される可能性がある

2.3.6. サインオフ

ドキュメントに対して必要なステークホルダーからのレビューと合意形成（サインオフ）を行う

ドキュメント作成担当とサインオフ担当

ドキュメント	作成担当	サインオフ担当
MRD	PO	他の PO メンバー（内部レビュア）、PM
PRD	PM	他の PM メンバー（内部レビュア）、PO, DEV リード、QA リード
DevSpec	DEV メンバー	DEV リード（内部レビュア）、PM、QA リード
TestSpec	QA メンバー	QA リード（内部レビュア）、PM、DEV リード

サインオフテーブルのサンプル　※PRD ケース

Role	PIC	Sign Off	Date	Comment

PO	<@担当者のメンション>	☑	19 Jun 2020	
PM	<@担当者のメンション>	☑	19 Jun 2020	
DEV	<@担当者のメンション>	☐		
QA	<@担当者のメンション>	☑	22 Jun 2020	

Role に求めるレビュー観点　※PRD ケース

Role	レビュー観点	Remark
PO	課題やユーザー要求を基に、ゴールや要件が適切に設定されているか	PO が MRD を作成した場合は、MRD の内容が PRD に落ちているかの観点で確認する
PM	PRD の内容に過不足はないか、レビュアにとって可読性があるか	PRD を作成したメンバーと異なるメンバーによる内部チェック
DEV	DevSpec を書くための情報が記載されているか	要件のトレーサビリティは取れるか、技術制限されていないか、など
QA	TestSpec を書くための情報が記載されているか	テスタビリティの情報は書かれているか（ログの取得の有無など）

レビュー依頼から合意までの流れ

レビュー方法

- 基本的にオンラインレビューとする。
- ドキュメントの重要性や情報の複雑性が高いとレビュイが判断した場合は、オンライン会議を開催する。

レビューの進め方

1. サインオフテーブルに Confluence の「アクションアイテム」機能を使用する。
2. レビュイは作成ドキュメントの編集ページ上のサインオフテーブルの各レビュア(ロール)の欄に、ショートカット（[ ]とキー入力）でアクションアイテムを作成し、'アクション'に「Sign-off request」、'@'でレビュア名、'//'で期限日を設定する。
3. レビュイは Slack のレビュー専用チャネルで、レビュアに対して概要ドキュメントのページを貼り、Sign-off 依頼を送る。
   ※ Slack はレビュアがレビュー依頼に気づいていないリスクの回避策
4. レビュアは Slack もしくは Confluence の「タスク」機能から、レビュー依頼がきていることを確認する。

5. レビュアはレビュー対象ページ上の「インライン」機能、もしくはページ下部のコメント欄でレビューコメントを記載し、「アクションアイテム」で設定された期限日までに「タスク」機能の完了ステータスを目指す。
6. レビュイはコメントに対して返信もしくは修正対応を行い、レビュアは不明点がクリアになったらレビュー対象ドキュメントのサインオフテーブルにサインオフ(チェック)する。

2.3.7. ドキュメントステータス管理

Jira’s KANBAN モード

ステータス		ステータスの説明	チケット操作
ISSUE	n/a	課題を起票した状態	• PO が起票 • 課題の概要、内容、起票者を明記する • PO が優先順位毎にチケットをレーンの上位に移動させる
PRD	In writing	PRD を作成中の状態	• PO が PRD を作成すべきチケットを ISSUE より移動する • PRD 作成後に、PRD のタイトルをチケットタイトルとし、同リンクをチケットに記載する
	In review	PRD のサインオフ待ちの状態	• PO が PRD レビューするべきチケットを In Writing より移動する • 移動時に Jira よりレビュアに通知メールが飛ぶ
	Signed off	PRD のサインオフ完了の状態	• PO が PRD レビュー後に合意（サインオフ）されたチケットを In Review より移動する • PO が優先順位毎にチケットをレーンの上位に移動させる • 移動時に Jira より開発メンバーに通知メールが飛ぶ
DevSpec	In writing	DevSpec を作成中の状態	• 開発チームが DevSpec を作成すべきチケットを PRD Signed-off より移動する
	In review	DevSpec のサインオフ待ちの状態	• 開発チームが DevSpec レビューするべきチケットを In Writing より移動する • 移動時に Jira よりレビュアに通知メールが飛ぶ

	Signed off	DevSpec のサインオフ完了の状態	• 開発チームが DevSpec レビュー後に合意（サインオフ）されたチケットを In Review より移動する • 移動時に Jira よりテストメンバーに通知メールが飛ぶ
TestSpec	In writing	TestSpec を作成中の状態	• テストチームが TestSpec を作成すべきチケットを DevSpec Signed-off より移動する
	In review	TestSpec のサインオフ待ちの状態	• テストチームが TestSpec レビューするべきチケットを In Writing より移動する • 移動時に Jira よりレビュアに通知メールが飛ぶ
	Signed off	TestSpec のサインオフ完了の状態	• テストチームが TestSpec レビュー後に合意（サインオフ）されたチケットを In Review より移動する • 移動時に Jira より PO に通知メールが飛ぶ
Done	Done	完了	• PO がチケットを TestSpec Signed-off より移動し完了とする
Pending	Pending	保留	• 各ステータスにおいて保留すべきチケットは、PO が移動させる

2.4. Product Backlog プロセス

PRD 作成以降の Agile/Scrum プロセス

2.4.1. 役割

Role	
PO	プロダクトオーナー
PM	プロダクトマネージャー
PMO	プロダクトマネジメントオフィス
SM	スクラムマスター
DEV	開発チーム
QA	テストチーム

2.4.2. プロセス

- PRD で何を作るかを明確にした後、DevSpec（開発仕様書）を作成し、どのように実装するかを記載する。
- DevSpec に記載する各機能は PRD のどの要件と紐付いているかを明確化する（トレーサビリティ）。
- DevSpec に対してステークホルダーがレビュー後同意する（サインオフ）。
- PRD の各要件を Product Backlog にストーリーチケット化する。
- スプリント開始前にバックロググルーミング MTG を実施し、バックログチケットを最新化する。
- スプリント開始日にプランニングを実施し、バックログチケットと DevSpec を技術の実現性を確認する。
- バックログチケットの優先度が高いチケットをポイントで相対見積もりし、スプリント期間に収まるタスクに細分化し Sprint Backlog に移す。
- スプリントを開始する。
- PRD 以降のプロセスで要件変更が発生した場合は、PRD を修正しサインオフする流れに戻る。

2.4.3. SPRINT Activity

2.4.4. MTGs

MTG	活動	タイミング
バックロググルーミング	• プロダクトバックログを見直し、見積もりや優先度の並び替えを行う（リファインメント） • PO, PM(PMO)が主体、必要に応じて開発リードが参加	スプリント開始前
スプリントプランニング	• プロダクトバックログを基に、スプリントバックログチケットを作成し、タスク内容や見積もり、担当を決定 • スプリントのゴールを決め、デモでどのようなアウトプットを出すかを決定 • スクラムマスターが主体、開発メンバーが参加、PO, PM(PMO)は任意またはオブザーバー	スプリント開始日
デイリースクラム（ハドル）	• YTB 形式で報告 　o Yesterday: 昨日実施したこと 　o Today: 本日実施すること 　o B: Blocker: タスクを妨げるもの • 短時間で実施（一人あたり 1-2min） • スクラムマスターが主体、開発メンバーが参加、PO, PM(PMO)は任意またはオブザーバー	スプリント期間中毎朝

スプリントレビュー（デモ）	• スプリントのアウトプットの報告 • 主に動くものを見せる • スクラムマスターが主体、開発メンバーがデモ実施、PO がプロダクトバックログを基に判断・承認する	スプリント最終日
レトロスペクティブ	• KPT&A 形式でスプリントの振り返り 　o Keep: 次も継続すること 　o Problem: 課題や問題 　o Try: 次に実施すべき施策概要 　o Action: 具体的な施策内容 • スクラムマスターが主体、開発メンバーが参加、PO, PM(PMO)はオブザーバー	スプリント最終日

Sign-Off

本ドキュメント(PRD プロセス)に対するレビューと合意（サインオフ）

Company	PIC	Sign Off	Date	Comment
<K 省>	<@担当者のメンション>	☐		
<A 社>	<@担当者のメンション>	☐		
<B 社>	<@担当者メンション>	☐		

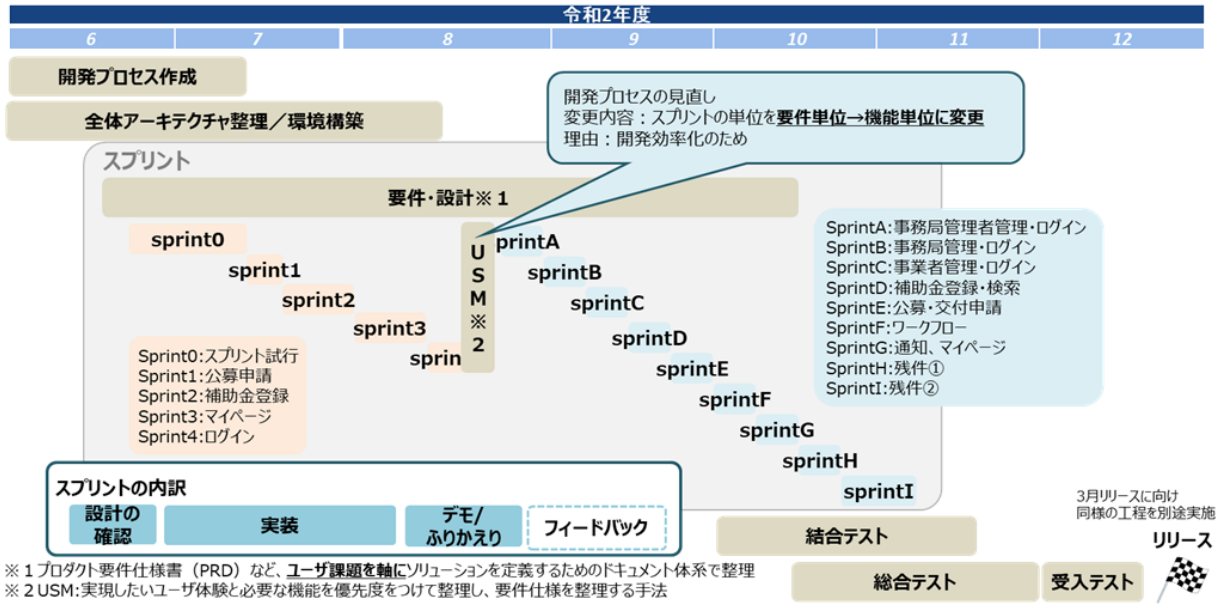
3. 実証結果

3.1. 実証の具体的な進め方

上記の 2.項に示した「Mini-WF と Agile のハイブリッド型のプロジェクト管理」の実証の具体的な進め方について、概要を以下に示す。

3.1.1. 実施スケジュール

実証対象とした開発プロジェクト（jGrants 2.0 の令和 2 年 12 月リリースを対象）のスケジュールを以下に示す。



3.1.2. 開発の進め方のポイント

開発の進め方の主要なポイントを以下に示す。

- **解決したい課題に優先度**をつけ、**優先度の高い課題に関わるユーザー要件から**設計・開発を行う。
- 1～2 週間のサイクル（スプリント、Sprint）を繰り返し、順番に機能を実装していく。1 つのスプリントの中では、システム実装からデモによる動作確認まで行い、**「使えるシステム」**となっていることを確認しながら開発を進めていく。
- 上記の進め方を実践する上で使用した PRD のイメージ例を以下に示す。

ホーム　最近　スペース　さらに表示　検索　共有

JGRANTSDEV2020 / ... / PRD*　4つのJIRAリンク

[PRD][1-1][事業者][補助金検索]補助金情報の一覧表示・検索・詳細表示

- Document
- Terminology
- Issue
- Goal
- KGI/KPI
- Scenario
- Scope
- Non-Scope
- Requirement
- 追加Requirement
- Detail
  - Day1認識合わせ資料
  - 【①補助金検索】
  - 【②補助金詳細確認】
- 追加Detail
- Release Dependency
- Open Issue
- Tenet
- Sign Off

Document

› List

Terminology

› List

Issue

- 行政が提供する補助金・助成金を俯瞰して検索できるサイトがなく、事業者は各省庁・自治体のサイト、民間が運営するサイト、商工会や支援サポーター、同じ業種のコミュニティなど複数から情報を集めないといけない状況である。よって行政はそれらの情報に触れていない事業者にリーチできていない
- 補助金の公募情報が個別の特設サイトなどに散在している現状では、事業者が自分に適した補助金を探すことが困難
- 申請手続きを行うサイトと公募情報や様式情報が公開されているサイトが分かれていると、事業者は申請作業を行う際に複数のサイトを行き来する必要があり不便

Goal

No	Goal	Priority
1	行政が提供する補助金・助成金の認知率を高める	P1
2	事業者が行政が提供する補助金・助成金の検索を一元的に行える	P1
3	事業者が自分に適した補助金・助成金に応募ができる	P1

KGI/KPI

- 補助金一覧ページへのPV数
- 補助金詳細画面のPV数
- 補助金申請へのコンバージョン数

Scenario

No	User scenario	Remark
1	A社はコロナ対策として工場含めた全事業所でオンライン会議するための環境を整備したいと考えたが資金が苦しく、政府あるいは自治体の補助金が受けられないかと思いついた。そこでgoogleで補助金と検索したところ1件目にjGrantsというサイトがひっかかり、表示したところ政府が提供する公式な補助金情報・申請サイトだとわかった。さっそく補助金検索画面を見てみると、「設備投資」「コロナ対策」という検索条件があったためそれらで検索したところIT導入補助金を見つけた。詳細ページを見たところ自社は要件を満たしておりまた申請に必要な書類も少なく整理そうであったため、早速申請をしてみることとした。	該当する補助金への迅速なリーチ
2	B社は政府あるいは自治体の補助金をたびたび活用していたが、制度については商工会議所に相談し情報を得ることが多かった。しかし、事前のリサーチや制度の比較は自分でも行いたいと考えていた。すると商工会議所からjGrantsというサイトを紹介され早速見てみると、政府・自治体の補助金が俯瞰して検索できいろんな制度の要件や支給上限額を見比べることができた。よって事前にjGrantsで利用できそうな制度に当たりをつけ商工会議所のアドバイスをもらうことで以前より効率よく採択を得ることができるようになった。	補助金公募情報への一元的なアクセス
3	C社は中小企業庁のものづくり補助金に数回応募しているが、採択にいたったことはなかった。今年もものづくり補助金の募集がはじまるため公募要綱を確認しようとjGrantsで「ものづくり」で検索したところ、地元の自治体でも同様の事業者をターゲットに補助を行っていることを初めて知った。詳細を確認したところ中小庁と変わらない支給額を受けることができ、また対象者	補助金申請機会の向上

Scope

- 補助金一覧の表示が適用される対象について以下に明記する
  - jGrants2.0にログインする前のシステム利用者
  - jGrants2.0にログインした事業者
  - jGrants2.0で公開設定されている補助金すべて

Non-Scope

- 補助金一覧の表示が適用されない対象について以下に明記する
  - jGrants2.0にログインした事務局・外部審査員
  - jGrants2.0で限定公開設定としている補助金全て
  - jGrannts2.0で非公開設定としている補助金全て

Requirement

› スプリント0

› old

ReqNo	Requirement	Priority	Goal No	Risk	Mitigation
1	事業者が補助金を簡易的に検索する				
1-1	事業者が、以下の項目を対象としてキーワード検索（部分一致）できる ・補助金番号 ・補助金名 ・補助金のキャッチコピー ・目的・概要 ・補助率 ・補助対象地域	P1	1		
1-2	事業者が、補助金の募集期間内かどうかで補助金を検索できる	P1	1		
2	事業者が補助金を詳細に検索する				
2-1	事業者が、補助金の実施機関（国、自治体等）で検索する	P2	1		
2-2	事業者が、地域、業種、金額で補助金を検索する	P2	1		
2-3	事業者は、英字で検索する場合大文字・小文字を区別せずに検索できる	P2	1		
2-4	事業者は、複数キーワードで検索することができる（区切り文字は全角スペースと半角スペースとする）	P2	1		
2-5	事業者は、検索項目を入力後にエンターキーを押すと、検索入力ボックスに入力した内容で検索できる	P2	1		
2-6	事業者が、補助金の支援の種類(事業者向けには「関心のある分野」と表示)を選択して補助金を検索する	P3	1		
2-7	事業者が、補助費用や補助率が高いものから順に補助金を見る	P3	1		
2-8	事業者が、自身の欲しい補助金を精緻に探せる程度の詳細検索を行う	P3	1		
2-9	事業者が、応募者が多い補助金の順に補助金を閲覧する	P3	1		
2-10	事業者が、類似の事業者の閲覧頻度の高い補助金を閲覧する	P3	1		
2-11	事業者が、閲覧している補助金に関連する補助金が表示されることを確認する	P3	1		
2-12	事業者が、補助金名が変更された場合でも同一の内容の補助金を見つけることができる	P3	1		
2-13	事業者が、キーワードの提案機能を使用する	P3	1		
2-14	事業者が、ドリルダウン機能を使用し、検索した補助金を絞り込む	P3	1		
2-15	事業者が、あいまい検索にて補助金を検索する	P3	1		
2-16	事業者は、自身の事業の目的を選択して、目的に適した補助金を一覧で確認で	P3	1		

追加Requirement

Req No	Requirement	Priority	Goal No	Risk	Mitigation	備考
7-1	(2021/1/27追記) 対応時期調整中 事業者は、補助金詳細画面から「一覧に戻る」で、一覧の元のスクロール位置＋検索条件を保持できる	P2	1			BTS-124から追加 https://metiproject.atlassian.net/browse/BTS-124

Detail

› old

Day1認識合わせ資料

【①補助金検索】

対象Requirement：
1-1.事業者が、以下の項目を対象としてキーワード検索（部分一致）できる
・補助金番号
・補助金名
・補助金のキャッチコピー
・目的・概要
・補助率
・補助対象地域
1-2.事業者が、補助金の募集期間内かどうかで補助金を検索できる
3-1.事業者は、補助金情報を一覧表示することができる
3-2.事業者は、検索結果の初期表示で登録日の降順で補助金を確認できる
3-3.事業者は、補助金一覧の情報を一定数ずつ確認でき、ページングで次の検索結果を参照できる
3-4.事業者は、補助金の検索結果をソートで並び替えることができる
3-5.事業者は、補助金の検索結果が0件の時は複質的に複制できる
3-6.事業者が、補助金の締切が近い順に検索結果を参照できる
3-7.事業者は、補助金一覧に表示された補助金を選択することで補助金の詳細情報を確認できる
3-8.事業者は、補助金一覧に表示された補助金が事務局によって非公開設定にされていた場合、補助金詳細画面に遷移できない
3-9.事業者は、補助金一覧に表示された補助金が事務局によって削除されていた場合、該当の補助金詳細画面に遷移できない

<①画面定義(イメージ)>

<2020/12/15追記>

キーワード検索(部分一致)は以下を対象とする
・補助金番号
・補助金名
・補助金のキャッチコピー
・目的・概要
・補助率
・補助対象地域

<2020/12/17追記>

検索した補助金の初期表示の表示順は「募集中の補助金のみ」のチェック有無によって変動する
・チェックなし：登録日の降順
・チェックあり：募集期間TO昇順

No	日付	対応種別	対応内容	ソース	更新
1	2020年9月16日	新規	-	Day1調整	ACN
2	2020年12月15日	追加	補助金名と目的・概要だけでなく、以下がキーワード検索の対象となるこ ・補助金番号 ・補助金名 ・補助金のキャッチコピー ・目的・概要 ・補助率 ・補助対象地域	BTS-107 https://metiproject.atlassian.net/secure/RapidBoard.jspa?rapidView=8&projectKey=BTS&selectedIssue=BTS-107&search=BTS-107	ACN
3	2020年12月17日	追加	検索した補助金の初期表示の表示順は「募集中の補助金のみ」のチェック有無によって変動する旨を追加	BTS-125 https://metiproject.atlassian.net/secure/RapidBoard.jspa?rapidView=8&projectKey=BTS&select	ACN

3.2. 振り返り（レトロスペクティブ）

3.2.1. 振り返りの契機

業務プロセス等の継続的な改善を目的とし、以下の契機にてプロジェクト参画者の間で振り返りを行った。

- スプリント0（スプリント試行）終了時 ⇒ 3.2.3.項を参照
- 各スプリント終了時 ⇒ 3.2.4.項を参照
- 12 月リリース向け開発終了時 ⇒ 3.2.5.項を参照
- 開発全体終了時（3 月リリース向け開発終了時） ⇒ 3.2.6.項を参照

3.2.2. 振り返りの方法

上記の契機ごとの振り返りは、以下の視点で行った（KPT 法）。

- **【K: To be kept】** What we should keep.（続けるべきことは何か）
- **【P: Problem】** Where we are having ongoing problems.（抱えている問題は何か）
- **【T: Try】** What we want to try in the next time period.（後続の期間に挑戦したいことは何か）

3.2.3. スプリント0（スプリント試行）終了時の振り返り

一連のスプリントを本格的に開始するのに先立ち、Sprint 0 によって PRD プロセスを試行した。Sprint 0 終了時の振り返りの結果を以下に示す。

(1)「K: To be kept」

No	To be kept
1	• スプリント 0 のデモで実際に動く機能を見ることで、それをベースに会話ができたのは素晴らしいと思う。
2	• プランニングでスプリントゴールを設定できたことで、デモをゴールと比較しながら見ることができた。
3	• スプリント 0 を体験したことで、今後のスプリントへの課題が見えた。今後もスプリントを通じて改善を図っていけると感じた。
4	• デイリーミーティング 進める上での懸念事項の早期把握や解決に有効だった。
5	• ユーザーストーリーをベースとした開発 前提を知らない開発者も、ストーリーを知ることである程度要望を理解できた。

6	• 短期間であったにもかかわらず、決めた Time-box に基づいてレビュー（Demo）及びレトロスペクティブが実施された点は、継続すべき素晴らしい点であると考える。

(2)「P: Problem」及び「T: Try」

No	Problem	Try
1	• PRD の詳細情報を詰める前にサインオフをしたため、DevSpec 作成時に不明点が発生した。	• PRD レビューを強化する。
2	• Jira の SPRINT0 のチケット（ストーリー、サブタスク）のステータスの変更がなかったため、現在の状況がわからなかった。	• 各チケットに担当を割り当て、担当にてステータスを設定する。
3	• スプリントプランニングで決めたストーリータスク以外を実施してしまうと、スプリント毎のチームの生産性が見えなくなってしまう（透明性の観点）。	• プランニングで決めたこと以外は実施しない。プランニング後に実施が必要となった場合は、都度 PMO<->Scrum Master 間でコミュニケーションしていく。
4	• 開発チームが何人で実施しているかが見えないと、スプリント活動を通じて、チームの生産性を理解していくことができない。	• スプリント活動を通じて、チームの生産性や成長を理解していく上で、プランニングで各ストーリーの相対見積もりや期間、人数などベロシティを測る要素を確認する。
5	• デモで動くものを見て終わりでは、今スプリントゴールの検証や改善課題が見えない。	• プランニングでスプリントゴールとデモの実施方法、デモで見る観点を決めておく。
6	• 要件の優先度の変更(P1->P2)が一部 PRD に反映されていなかったため、タスクスコープが不明瞭となった。	• 今後プロダクトバックロググルーミング(リファインニング)で優先度の認識合わせを行う。
7	• スプリント途中で想定外のタスクが発生しても、PMO はデモまで把握していなかった（コミュニケーション観点）。	• スプリントの途中で追加や問題が発生した場合は、都度コミュニケーションをとり、必要あれば方針を変えていく。
8	• スプリント活動自体はまだ改善の余地があると感じた。	• 次回のスプリントについても、短期間として早めの振り返りを実施していく。

9	• 事前の追加検討が必要となり、当初想定の作業以外の検討が必要となった。 • 開発環境準備・改善 • 画面デザイン検討 • データモデルの検討 • システム処理方式の検討	• 業務アプリケーションの基礎となる部分（データモデル、システム処理方式、共通化事項）の先行実施が必要。 • 全体観をもち、上記を踏まえ構築順番の検討が必要。
10	• PRD （Requirements）の観点に偏り、漏れがないか確認が困難であった。	• 標準実装要件ガイドライン(一覧)を作成し、レトロスペクティブで改善していく。

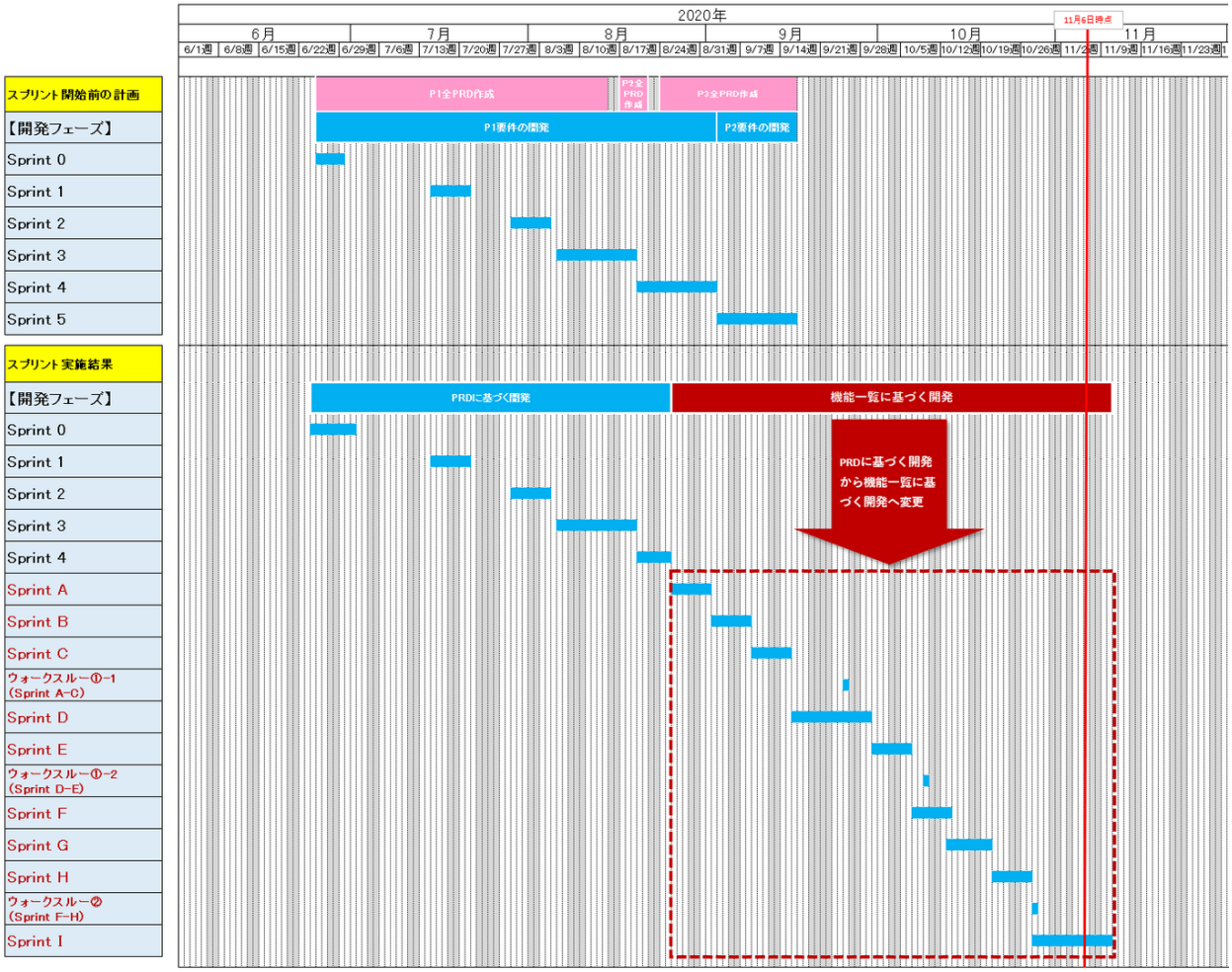
3.2.4. 各スプリント終了時の振り返り

各スプリントの終了時には振り返りを実施した。各スプリントで挙げられた「K: To be kept」及び「P: Problem 及び T: Try」については、3.2.5(3)項のインペディメント・リストの中にまとめて示した。

3.2.5. 12 月リリース向け開発終了時の振り返り

令和 3 年 3 月リリース向けのスプリント群を適切に計画することを目的とし、実施済の令和 2 年 12 月リリース向けスプリント群の計画変更の履歴を以下のように整理した（令和 2 年 11 月 6 日時点の状況に基づく）。

(1)スケジュールの変更履歴



(2)計画変更等の概要

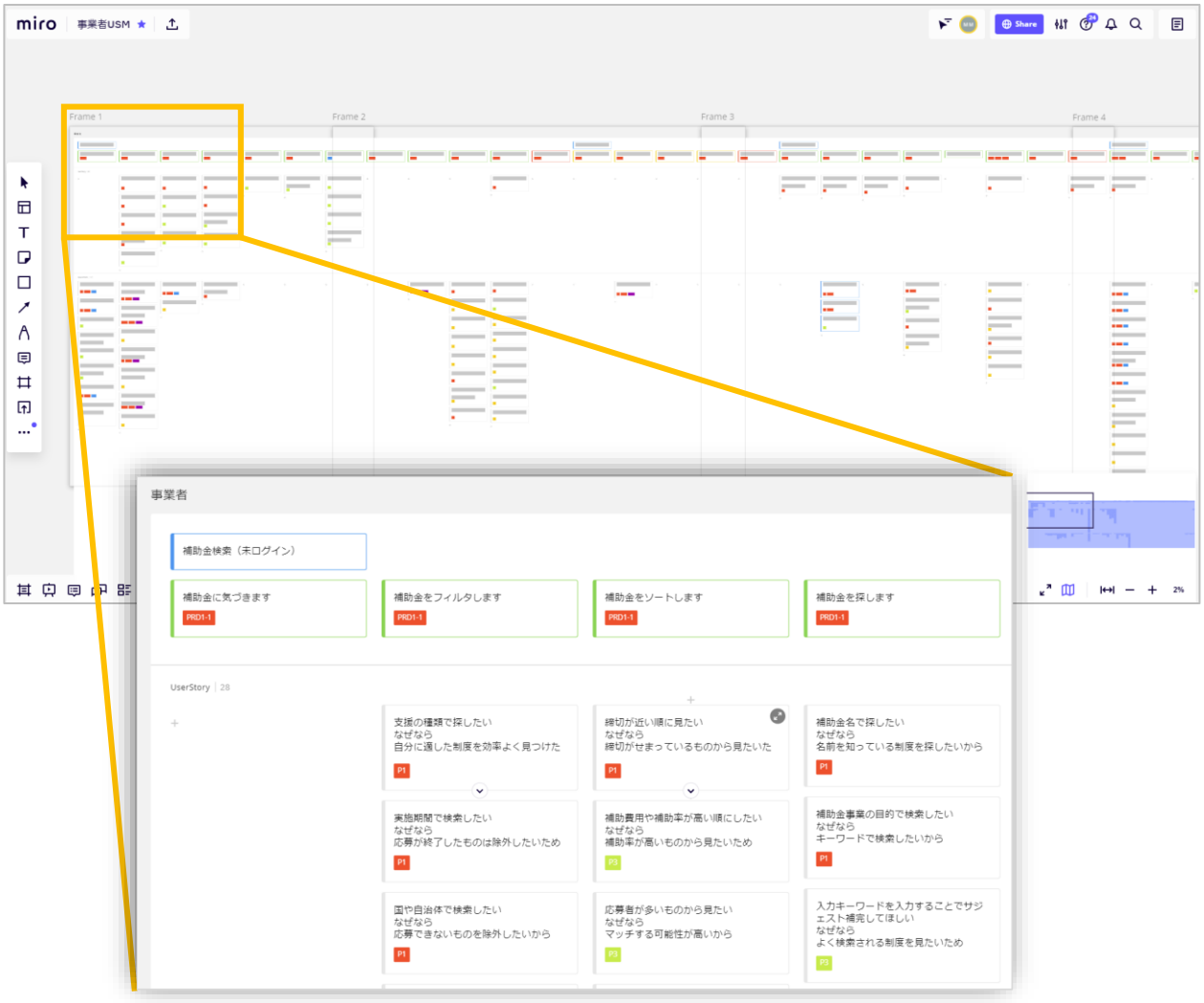
No	対象スプリント	期間	計画変更等の概要
1	（スプリント開始前）	―	• 「Mini-WF と Agile のハイブリッド型」の採用を決定。 • PRD プロセスで試行的に Sprint0 を回すことにした。
2	Sprint 0	24 Jun 2020 01 Jul 2020	• Sprint 0 の結果のデモを初めて受ける。Sprint0 の期間だけでできたものではなく、あらかじめ「デモ」として作り始めていたもの（PRD 外の機能も実装）であった。
3	Sprint 1	15 Jul 2020 21 Jul 2020	• Jira ボード管理に取り組む。
4	Sprint 2	29 Jul 2020 04 Aug 2020	• PRD のドキュメンテーションの負荷が大きいこと、機能の全量が知りたいとの声あり。 • 機能ではなくユーザー体験の全量を出すべきとのアジャイルコーチからのアドバイスに基づき、ユーザストーリーマッピング（USM）の作成に着手した。
5	Sprint 3	06 Aug 2020 19 Aug 2020	• 開発事業者側で開発技術者（リーダークラス）を要件・仕様調整窓口として設置した。 • PRD とは別に「機能要件一覧」が登場した。 • 経済産業省側で触れるシステム環境が公開された。 • PRD 執筆ボトルネックが顕在化した。 • 開発事業者より、機能一覧で開発スコープを決めたい、構築ドメインごとに開発を進めたい（業務フローごとでは開発効率が悪いため）との意見あり。⇒Sprint 4 は予定通り実施しつつ、新たなスプリント計画を立案することとした（以下の Sprint A～I）。
6	Sprint 4	20 Aug 2020 25 Aug 2020	• USM ワークショップを実施した。 • 新たなスプリント計画（ドメイン単位）を議論した。 • 1 週間のスケジュールで以下のように進めることとした。 day1: 基本設計程度の要件決定（80%くらい決める） day2-4: 実装 day5: デモ及びレトロスペクティブ
7	Sprint 5	実施せず	• スプリント計画の変更に伴い中止。

8	Sprint A	26 Aug 2020 01 Sep 2020	• 12 月リリース向け開発終了が、当初予定の 9 月中旬には間に合わない見込みとなり、マスタースケジュール変更に着手。 • 複数のウォークスルーを設け、各ウォークスルー時点までに開発完了した機能を確認後にドメイン単位で結合テストを進める方針となった。
9	Sprint B	02 Sep 2020 08 Sep 2020	• アジャイルイベントを1日（水曜日）にまとめて実施することとした（当面は対面にて）。 • 10 月から開発要員を減少させる計画であったが、1 月まで維持することに変更された。
10	Sprint C	09 Sep 2020 15 Sep 2020	―
11	ウォークスルー①-1	25 Sep 2020	• Sprint A、B、C を対象としてウォークスルーを実施。 • ウォークスルーは業務要件との整合確認が目的であり、PRD との整合はこの場で確認することとした。
12	Sprint D	16 Sep 2020 29 Sep 2020	• 祝日をはさむためスプリント期間を 1 週間ではなく 2 週間とした。 • 10 月中旬までのスプリントで全機能開発できないリスクが顕在化し対策されていない可能性を課題として識別した。
13	Sprint E	30 Sep 2020 06 Oct 2020	• SalesForce による機能制限などが顕在化（アクセスコントロールなど）。 • UI デザインは後回し、まず機能開発優先で進めた。 • 開発効率化のため、打合せ参加メンバーを絞り、また、開発拠点を集中化した。
14	ウォークスルー①-2	09 Oct 2020	• Sprint D、E を対象としてウォークスルーを実施。 • UI デザインは、Spring G あたりで取り組むこととした。
15	Sprint F	07 Oct 2020 13 Oct 2020	―
16	Sprint G	13 Oct 2020 20 Oct 2020	• バックログの残ストーリーポイントが多く、総合テストを 2 回に分割し、11/10 まで 2 スプリントを追加で実施することを決定した。
17	Sprint H	21 Oct 2020 27 Oct 2020	• ユーザーテスト開始を 11 月 2 日から 11 月 9 日へ延期した。 • アジャイルイベント、Jira チケット管理を省略した。

18	ウォークスルー②	28 Oct 2020	• Sprint F、G 、H を対象としてウォークスルーを実施。
19	Sprint I	28 Oct 2020 10 Nov 2020	• 一連のスプリントによる開発を完了。

【特記事項】

- Sprint 2 では、機能ではなくユーザー体験の全量を出すべきとのアジャイルコーチからのアドバイスに基づき、ユーザーストーリーマッピング（USM）を作成し、ユーザー体験に基づく開発を行う方針へ変更した。ユーザーストーリーマッピング（USM）のイメージ例を以下に示す。



- 開発事業者より、機能一覧で開発スコープを決めたい、構築ドメインごとに開発を進めたい（業務フローごとでは開発効率が悪いため）との意見があり、その方針変更を踏まえた新たなスプリント群（Sprint A～I）への計画変更を行った。

(3)インペディメント・リスト

インペディメント・リストは、スクラムによるシステム（ソフトウェア）開発において、開発チームの円滑な開発活動を阻害する要因（Impediment）を一覧化したリストである。インペディメント・リストによる管理の目的は、スプリントレトロスペクティブで識別・累積された問題（Problem）、解決試行（Try）、アクション（Action）等について、重要なものを中心に問題解決の状況を追跡管理（トレース）することである。

12 月リリース向け開発終了時の振り返りでは、開発活動の阻害要因（Impediment）と問題（Problem）について、原因と結果（Cause and Effect）の関係にあるものとして捉え、jGrants2.0 開発のスプリントレトロスペクティブで識別・累積された問題群（Problems）を阻害要因（Impediment）の観点でカテゴライズし、整理及び管理することとした（以下の表を参照）。

開発活動の阻害要因（Impediment）	対象スプリント《実施期間》	スプリントで発生した問題（Problem）	解決試行（Try）／アクション（Action）	状況（Status）	優先度（Priority: P1, P2, P3）
ユーザー要件・設計仕様の妥当性確認の不足	Sprint 0《6/25-6/29》	PRD（Requirements）の観点に偏り、漏れがないか確認が困難であった。	標準実装要件ガイドライン（一覧）を作成し、レトロスペクティブで改善していく。		
	Sprint 2《7/23-8/5》	V1 ベースの仕様にまとまりがちである。	N/A	今回のアーキテクチャを前提とした Requirement の実現方式について、DAY1(プランニング)時に提案させて頂いている。ただし、V1 の仕様を踏襲しなければならない Requirement も一部あるため、その都度認識合わせを実施。	P1
	Sprint G《10/21-10/28》	開発当初から懸念していた UI/UX のブラッシュアップタイミングがユーザーテストに間に合わなかった。特にステークホルダーからの声で急な対応を余儀なくされた。	プロジェクト管理上、ユーザビリティを改善するタイミングとしてのマイルストーンを明確におく。また、早い段階から DesignSpec などのドキュメント、またはプロトタイプツールなどを活用して、ステークホルダーと早い段階での UI/UX の合意形成を図る。セカンドリリースでは、ぜひ意識して進めたい。	3 月向けには設計開発とは別に UI/UX を検討するチームを設ける。	P1

ユーザ受入の妥当性の不明瞭	Sprint 0《6/25-6/29》	デモで動くものを見て終わりでは、今スプリントゴールの検証や改善課題が見えない。	プランニングでスプリントゴールとデモの実施方法、デモで見る観点を決めておく。	【Keep】 ゴールが達成できているかの検証のために、シナリオを作成しデモを実施。ただし、時間に限りがあるため、デモでは主要シナリオの確認を行い、その後 1 週間のフィードバック期間を設け、経済産業省での確認を実施。	P1
開発ベロシティの不明瞭	Sprint 0《6/25-6/29》	スプリントプランニングで決めたストーリータスク以外を実施してしまうと、スプリント毎のチームの生産性が見えなくなってしまう（透明性の観点）。	プランニングで決めたこと以外は実施しない。プランニング後に実施が必要となった場合は、都度 PMO<->Scrum Master 間でコミュニケーションしていく。	【Keep】 現在は、DAY1(プランニング)で決めた内容を基に開発を実施している。実現方法について相談が必要な場合は、適宜共有しながら進めている。	P1
	Sprint 0《6/25-6/29》	開発チームが何人で実施しているかが見えないと、スプリント活動を通じて、チームの生産性を理解していくことができない。	スプリント活動を通じて、チームの生産性や成長を理解していく上で、プランニングで各ストーリーの相対見積もりや期間、人数などベロシティを測る要素を確認する。	Sprint A～G のベロシティについて別途整理のうえ報告させて頂く。なお、3 月向けには、これまでのベロシティと 3 月向け機能のストーリーポイント（概算）を基にスケジューリングする。	P1

	Sprint 3《8/6-8/20》	スプリント毎のベロシティをもう少し精緻に理解したい。	スプリントでの開発体制や人員数、各自の役割や稼働日などをプランニング時に全員で認識共有し、グルーミング（リファインメント）時やレトロスペクティブ時に実績を共有する。	Sprint A～G のベロシティについて別途整理のうえ報告させて頂く。なお、3 月向けには、これまでのベロシティと 3 月向け機能のストーリーポイント（概算）を基に、スケジューリング予定。	P1
	Sprint 3《8/6-8/20》	今回から 2 週間スプリントでしたが割り当てた PRD は 3 枚だったので、当初はスプリントの途中でスコープを拡大できることを期待していた。	DevSpec の取り扱い変更やチーム分割案など開発スピードアップ策を議論さしたが、それによりスピードアップに寄与できそうか逐次状況を確認する。		
スプリントに係るプロセスの不明瞭	Sprint A《8/27-9/1》	会議調整が不十分なままデモを迎えてしまった。	再度会議体の建付け含め整理・議論の上、決定する。		
	Sprint A《8/27-9/1》	プロセス、アジャイルイベントの方法、日程などが決まっていないままスプリントの終了を迎えてしまった。	アジャイルイベント（日程、場所、方法）、コミュニケーション（コメントやフィードバックの仕方、Demo の仕方等）はどれを維持しどれを見直すのか、しっかり定義する。		

リスク管理の不明瞭	Sprint F《10/7–10/13》	スプリント内外問わず、様々リスクが顕在化してきているが、リスクへの具体的な対応方向性が見えないため不安が大きい。リスクへの手当が遅れている一因として考えられるのが、課題とリスクで管理を分けていることも考えられる。	リスクを課題に落としこむタイミングをどうするかを明確に決めておく、もしくは課題チケットに一本化し、優先度のみで課題とリスクを表現するのはどうか。	以下の進め方を検討。 ①週次でリスク検討会を定例枠で実施し、アクションの優先度を協議する。（この時点ではリスク管理簿で管理） ②アクションが必要となったものは、チケット化し、他の OpenIssue と合わせてタスク管理を行う。	P1
情報の分散	Sprint B《9/2–9/9》	UAT で過去デモのフィードバックをする場合、現スプリントの Jira チケットや Confluence の Feedback 欄だと管理しづらい	後のテストフェーズで Issue のチケット管理されるのであれば、今のうちに Jira KANBAN ボードで管理しておく。	【Keep】 UAT やデモで検知した不具合は Issue として管理／要望はバックログとして管理する運用を実施。	P1
	Sprint C《9/10–9/16》	デモ後の動作確認で利用できる環境やそのための条件、要件との突き合わせなどの情報が Slack や Backlog, Sharepoint, Confluence で分散しているため、情報へのアクセスや最新情報の把握に手間がかかる。	該当スプリントデモで使用した画面やドキュメントへのアクセス情報はこのページの Related Information に記載する。 スプリントにまたがるような情報は、Tips として Confluence に情報を集約する。	Slack に点在する情報の扱いについて検討。	P1

	Sprint E《9/30–10/6》	Sharepoint と Confluence でドキュメントが分散してしまっているため、情報へのアクセシビリティが悪い。	ドキュメントツールの統一化が現状厳しいため、情報の質により住み分けしてはどうか。例えば、決定した情報や高頻度で参照する情報は Confluence（納品観点やアクセスしやすい観点から）、議論中の要件や仕様は Sharepoint など。	【Keep】 左記の方針を踏まえ、現在は、決定した情報や高頻度で参照する情報は Confluence（PRD、DevSpec 等）、中間成果物（検討資料等）は Sharepoint で管理。	P1
情報の共有（可視化）・更新の不足	Sprint C《9/10–9/16》	ArchiSpec や他のドキュメントのアップデートの状況が見えない。	ArchiSpec, DevSpec, DesignSpec などのドキュメントを介した議論の必要性を検討し、必要なものはスケジュールにて明確化する。	【Keep】 設計に関する議論が必要な場合は、DAY1(プランニング)時やスプリント外定例にて議論を実施。	P1
	Sprint E《9/30–10/6》	Sharepoint と Confluence でドキュメントが分散してしまっているため、情報へのアクセシビリティが悪い。	ドキュメントツールの統一化が現状厳しいため、情報の質により住み分けしてはどうか。例えば、決定した情報や高頻度で参照する情報は Confluence（納品観点やアクセスしやすい観点から）、議論中の要件や仕様は Sharepoint など。	【Keep】 左記の方針を踏まえ、現在は、決定した情報や高頻度で参照する情報は Confluence（PRD、DevSpec 等）、中間成果物（検討資料等）は Sharepoint で管理。	P1

開発作業状況の可視化の不足（チケット）	Sprint《6/25-6/29》	Jira の SPRINT0 のチケット（ストーリー、サブタスク）のステータスの変更がなかったため、現在の状況がわからなかった。	各チケットに担当を割り当て、担当にてステータスを設定する。	【Keep】 1 つのチケットを複数名で開発担当しているため、管理者がチケット管理を実施している。	P1
	Sprint 1《6/30-7/22》	スプリント中に新たに発生、発覚した課題や問題がスプリントの最終日になるまで共有されない。	完全に開発チーム内に閉じた問題（他チーム、ステークホルダーに全く影響がない事柄）や即時で解決する課題以外は、積極的にステークホルダーとコミュニケーションする。	【Keep】 開発チームと仕様調整チームで日々MTG を実施しており、経済産業省と調整が必要な課題については、Slack でコミュニケーションを実施。	P1
	Sprint 3《8/6-8/20》	PRD_MANAGEMENT（KANBAN Jira）のステータスが実際の進捗と乖離しているので、チケットベースでの状況把握が難しい。	今一度、誰が全チケットをコントロールするのか、ステータス毎に主体が変わるのかを認識合わせして、精緻に状況が見られるようにしておく。	機能一覧ベースでの対応になったため、PRD_MANAGEMENT は使用していない。現在、PRD の更新を実施しているが、Slack 経由でのレビュー依頼になっている。	P1
	Sprint A《8/27-9/1》	Jira（PRD_MANAGEMENT、スプリント共）の更新がされていないため状況が見えない。	前回の KPT 同様、ハドルなどを利用して Jira のアップデートを行う。	【Keep】 現在のチケットは、担当者レベルのタスクまで細分化されていないため、スプリント期間の後半でステータスが変更される見え方となっているが、管理負荷軽減のために現状の運用を継続させて頂きたい。	P1

	Sprint B《9/2–9/9》	カンバンボードに「DEMO準備完了」「確認中（UAT）」を足してもらったと思うがいったん構築が終わったら Done としないと進捗が見えない。	作業が終わったら Done でよいのではないか。		
	Sprint C《9/10–9/16》	ログアウトはまだ gBizID との調整が残存しているが完了となっており、残件に何があるのかわからなかった。	残件があるのならチケット上 Backlog に残しておきたい。		
開発作業状況の可視化の不足（バーンダウンチャート） • バーンダウンチャート・バーンアップチャートの使い方のノウハウ不足	Sprint 1《6/30–7/22》	各種レポート(バーンダウン、バーンアップ、ベロシティ)はストーリー単位で計測される。	今後本レポートを利用するために、起票方法を変更すべきか相談する。		
	Sprint 2《7/23–8/5》	スプリントバーンダウンチャートが機能していない。	理想線が表示されるようにスプリント開始前にチケットに見積もりを入れる。		
	Sprint 4《8/21–8/26》	スプリント期間が 2week → 1week に短縮されたが、バーンダウンチャートの期間が 2week のままとなっている。	Jira チケットは常に最新状態を保てるようにデイリーハドルなどをうまく活用してアップデートする。		

• ツール使用ルールの不明瞭 • 問題発生時の調整ルールの不明瞭 • 管理作業の負荷	Sprint C《9/10–9/16》	スプリント C のバーンアップチャートで、途中からストーリーポイントが2倍に増えている。	プランニングで決定した事項から変更があった場合は、都度 Slack や会議を利用して情報共有する。	【Keep】 機能の分割などの必要性によりチケットが増えたことで、ストーリーポイントが増えてしまった。現在は、プランニングで決定した事項から変更があった場合は、都度情報連携しながらすすめている。	P1
	Sprint B《9/2–9/9》	ストーリーが Done になっていないのでスプリントバーンダウンチャートで消化状況がわからない。	1と関連するがタスクが終わったら Done としすべてのタスクが Done となったらストーリーも Done でよいのではないか。		
	Sprint B《9/2–9/9》	バーンダウンチャートで、なにか最後のほうにストーリーポイントが跳ね上がっている。	追加タスクが見つかったのか単位反映漏れか確認する。		
	Sprint B《9/2–9/9》	バーンアップチャート上は、プランニング時より 10pt 近く減っているので、最終的にチケット作業量が少なくなったように見える。	チケット上のポイント付け替えであれば問題なく、何か問題が起きたのであれば共有する。		
	Sprint C《9/10–9/16》	スプリント C のバーンダウンチャートがないため状況が見えない。	スプリント開始直後にチャートが表示されることを確認する。		

	Sprint D《9/17–9/29》	バーンダウンチャートが落ちていない。	引き続きスプリント期間中のチケット更新。		
	Sprint E《9/30–10/6》	スプリントの最終日の時点で、バーンダウンチャートが総数 150pt のうち 100pt 残となっていました。	各ストーリーチケットが単体テストを完了しないとクローズできないなど、もしクローズ条件に共通したブロッカーがある場合は、共通タスクをまとめて別チケット化するのも一案。		
開発作業状況の可視化の不足（CI／CD）	Sprint B《9/2–9/9》	CI/CD パイプラインとテスト自動化がすでに導入開始されているので、テストやデプロイ状況や結果を見られるようにしてほしい。	プランニング時にスプリントでどんな取り組みをするのか、デモ時に数値やグラフなどで可視化して、どのような状況や結果だったかを共有されると、今後他のテストフェーズでの取り組みのインプット材料になる。	スプリント開発のテストをパスしたものをデプロイしている。	P1
開発作業状況の可視化の不足（スプリント外） • スプリント以外の活	Sprint C《9/10–9/16》	スプリント外ではあるが、ワークフローの検討やウォークスルーへ向けた開発も走っており、Jira での管理が難しい。（スプリントにおける管理という意味では役に立っている）	スプリントの管理という意味で利用を継続するか検討。		

動状況の可視化の仕組みが不足	Sprint C《9/10-9/16》	デモ画面にアクセスしたところ、事業者の申請画面もデモのタイミングからアップデートが入っているように見えた。スプリント以外の開発の実態がわからない。		スプリントデモの環境は、開発環境のため、開発状況に応じて更新がかかる。	P1
ユーザー要件に係るコミュニケーションの不足 • PO-開発の間のコミュニケーション不足 • 開発T内のコミュニケーション不足	Sprint 0《6/25-6/29》	PRD の詳細情報を詰める前にサインオフをしたため、DevSpec 作成時に不明点が発生した。	PRD レビューを強化する。	3 月向けには PRD/Backlog の棚卸を実施したうえで、開発着手予定。	P1
	Sprint 0《6/25-6/29》	要件の優先度の変更(P1->P2)が一部 PRD に反映されていなかったため、タスクスコープが不明瞭となった。	今後プロダクトバックロググルーミング(リファインニング)で優先度の認識合わせを行う。	【Keep】 Sprint 対象の確認を DAY1(プランニング)で実施。	P1
	Sprint 0《6/25-6/29》	スプリント途中で想定外のタスクが発生しても、PMO はデモまで把握していなかった(コミュニケーション観点)。	スプリントの途中で追加や問題が発生した場合は、都度コミュニケーションをとり、必要あれば方針を変えていく。	【Keep】 開発チームと仕様調整チームで日々MTG を実施しており、経済産業省と調整が必要な課題については、Slack でコミュニケーションを実施。	P1

	Sprint 1 《6/30- 7/22》	DevSpec 作成の際に、PRD の修正追記の必要性に気づく可能性がある。 • Requirement の追加について 【問題点】「事業者は入力内容を確認できる。」という要件に対し、「申請の新規作成⇒プレビュー」と「一次保存した申請の再編集⇒プレビュー」の 2 つの観点について記載ができておらず、1 つの Requirement になっていた。 【対処の方法】DevSpec 上で内容を分割して記載した（⇒Requirement の追加が発生） • PRD の Detail における記載からの変更 【問題点】実装における制御方針について、PRD の Detail で記載したものから、DevSpec における検討の中で変更が望ましいという結論になった	すべてを PRD 作成の際に検知するのは難しい認識、後続タスクで発見した際にどうすべきかを決めておく必要がある。	12 月向け開発では開発を優先し、PRD の修正の優先度を下げた。そのため、現在、3 月向け開発のために PRD の最新化を実施。 3 月向け開発の進め方と併せて、ドキュメント更新タイミングを協議させて頂きたい。	P1

		【対処の方法】DevSpec 上で PRD の Detail に記載した内容から変更した旨をコメントで記載した			
	Sprint 2《7/23-8/5》	DevSpec が Sprint 期間内にサインオフまで完了できていない。	PRD 作成を前倒し、DevSpec の作成、レビュー時間の確保ができるようになるようできる限り今後はスケジュールを調整していく。	3 月向けには PRD/Backlog の棚卸を実施したうえで DevSpec を作成し、開発着手予定。	P1
	Sprint 2《7/23-8/5》	議論の深堀ができていない。PRD の執筆やサインオフ、スプリントに入ることが最優先となっており OpenIssue に課題がたまっている。	大きな論点は PRD のコメントレベルではなく別に時間を設けて議論をすべきではないか。	【Keep】 Sprint 外定例にて、課題検討を実施。	P1
	Sprint 3《8/6-8/20》	一部 PRD で確認漏れがあり(CSV 出力項目)、Sprint が始まってからの確認となってしまった。	より開発 T と蜜に連携をとり、PRD 作成の段階で検知できるようにする。	【Keep】 頻度や方法（Teams を用いて証跡を残す等）を見直し済み。	P1
	Sprint A《8/27-9/1》	AS-IS の確認が不十分だったため、業務の確認がデモ後となってしまった。	AS-IS の確認を意識して DAY1 へ向けた準備を行う。		
	Sprint A《8/27-9/1》	一部の PRD 項目がサインオフされないまま、スプリントが開始・終了している。	前スプリント期間で次のスプリントで何をやるかのリファインメントを実施する。		

	Sprint C《9/10–9/16》	大まかなユーザーストーリー、画面構成は決まってきているが項目詳細仕様、デザインなどは問題ないか懸念がある。(指摘や現状で問題なければ問題なし)	もしご指摘などがあれば Feedback 欄にいただく。	3 月向けには設計開発とは別に UI/UX を検討するチームを設ける。	P1
	Sprint B《9/2–9/9》	DAY1 の確認の中で弊社と本省の業務認識齟齬が発覚した。	認識を擦り合わせるため、主要論点については対面での会議を設定。	【Keep】 Sprint 外定例にて、課題検討を実施。	P1
	Sprint D《9/17–9/29》	データモデルなど実装設計の議論をする際に、前段としてどのような要件に基づいているのか、シナリオに紐づいているのかなどが見えない場合の懸念として、指摘しづらいまたは指摘時に要件の話などに戻ってしまい時間がかかる、要件に精通してないメンバーが話に入りにくいなどがありうる。	該当する PRD#や Requirement#などを先に指定する、話が他の要件に及ぶ際は途中で提示するなどの情報補完を実施する。	【Keep】 打合せ時に、関連する PRD を示しながら進めるなど、会議運営の工夫を実施。	P1
	Sprint E《9/30–10/6》	リモートサイトにおけるメンバーとの認識齟齬が翌営業日に発覚した。	DAY1 で実施した作業を踏まえた内容を開発メンバーで整理・共有する場を DAY1 当日にセットする。		

ユーザーサポートとのコミュニケーションの分離	Sprint D《9/17-9/29》	補助金マスターの設定項目について、併行してユーザーサポート側が実施している整理（支援カテゴリー×事業分類）が取り込む段取りが必要と思われました。	スプリントプラニング、レビューで「ユーザーサポートからの連携事項」（無ければ無いこと）を確認する。		
外部インタフェース先に係るリスク	Sprint 4《8/21-8/26》	gBizID のような外部システムは jGrants 側でコントロールできないため、開発やテストに遅れを生じさせるリスクが比較的高い。	スプリント開発においては、接続先システムの代替となるスタブやドライバーを導入して単体テストを実施する。	【Keep】 gBizID は実際のものを利用して開発やテストを実施中。今後、実際の者が使用できない場合は、スタブやドライバーを用いた検証を実施。	P1
作業計画の無理	Sprint 1《6/30-7/22》	デモで見せて頂いた画面や機能は実際に本番運用で使えるサービスレベルではない。	今はスプリント自体の試行期間でもあることは理解していますので、今後のスプリントでは実際にリリース可能なレベル（総合テストでの品質保証レベルではない）でデモができること。		
	Sprint D《9/17-9/29》	Sprint の Day1 で発生した宿題事項に対し、全てを Sprint 中に社内対応・経済産業省の確認・実装まで完了させることが期間的に難しい。	①Sprint の Day1 までに、slack 等で事前調整を行う。 ②Sprint の Day1 で発生した開発事業者持ち帰り事項については、優先度に応じて Sprint 内対応か次回 WT まで対応か決定する。	原則 1 週間サイクルでスプリントを進めていたが、今後は開発対象のストーリーポイントを基に期間を変動させる、もしくはバックログで管理する運用を実施する。	P1

	Sprint D《9/17-9/29》	覚悟はしていたがウォークスルー対応と Sprint 対応の並走の負荷が高かった。	シナリオの簡素化などの工夫を検討する。	要員追加の他、会議運営の見直し、ドキュメントの簡素化、作業場所の工夫、開発優先度の見直し等を実施。 3 月向けには、これまでのベロシティと 3 月向け機能のストーリーポイント（概算）を基に、スケジューリングする。	P1
	Sprint E《9/30-10/6》	ベロシティが 1 週間で 80 程なのに対し、スプリントプランニングの段階でストーリーポイントが 150 程割当たっていた。それに対し明確な対策もなく開発者の負担が高い。	体制増強など手は打っていただいている理解であるが、効率化のための工夫（仕様簡素化、コミュニケーションの効率化、ドキュメントの体裁簡素化、など）と、それらをオープンにする。	要員追加の他、会議運営の見直し、ドキュメントの簡素化、作業場所の工夫、開発優先度の見直し等を実施。 3 月向けには、これまでのベロシティと 3 月向け機能のストーリーポイント（概算）を基に、スケジューリングする。	P1
	Sprint F《10/7-10/13》	Sprint における開発工数が不足している。	対応策を引き続き講じる、また新たに対応可能な策があれば経済産業省に相談する。 -会議参加者を絞る -ドキュメントの簡素化 -拠点における開発 -ホットライン(経済産業省に相談） 等		

	Sprint F《10/7-10/13》	前スプリントと同じProblem だが、開発チームのベロシティ以上のタスクが割り当たっている。	リスクチャンネルでもコメントしたが、リソースの集中や周辺タスクの簡素化などで解決できない次元にきていると感じる。リカバリスケジュールやリスク管理強化などのマネージメントを強化してほしい。	以下の進め方を検討中。 ①週次でリスク検討会を定例枠で実施し、アクションの優先度を協議する。（この時点ではリスク管理簿で管理） ②アクションが必要となったものは、チケット化し、他のOpenIssue と合わせてタスク管理を行う。	P1
	Sprint G《10/21-10/28》	1 週間のスプリントではイベントへ向けた準備も必要となり、期間内でできることが限られる。	SprintG で実施させていただいたように、内容量に応じてイベントを簡略化する、期間の調整をさせていただくなどが必要。		
	Sprint G《10/21-10/28》	1 週間のサイクルを回し続けるのはレビューする立場としても辛いものがあった。イベント日が多く次のイベントに向けた準備が間に合わない。	2 週間のスプリントを挟むなど、緩急つけたスケジュールが組めると負担なく続けられる。		

開発環境の準備の手間取り	Sprint 0《6/25-6/29》	事前の追加検討が必要となり、当初想定の作業以外の検討が必要となった。 • 開発環境準備・改善 • 画面デザイン検討 • データモデルの検討 • システム処理方式の検討	• 業務アプリケーションの基礎となる部分（データモデル、システム処理方式、共通化事項）の先行実施が必要。 • 全体観をもち、上記を踏まえ構築順番の検討が必要。	12 月向け開発では、PRD ベースでの開発プロセスでは、システムの全体整合性が確保されず、全体アーキ/データモデルを固めるべく機能一覧ベースでの開発プロセスへ変更した。 3 月向け機能の開発プロセスについて、以下の方針で検討。 ・外部 API、各種手続きは、ウォーターフォール（PRD 作成次第、開発範囲を協議しスケジューリング） ・12 月向け機能改善やユーザーテスト要望は、アジャイル（開発期間はバックログのストーリーポイント/ベロシティを基に設定） ・UI/UX 改善は別体制で実施。	P1
	Sprint 4《8/21-8/26》	gBizID との調整に時間を要した(必要な申請・質問対応含む)。	必要な Sprint に間に合うよう調整が必要。		
スプリントイベントの非効	Sprint 0《6/25-6/29》	スプリント活動自体はまだ改善の余地があると感じました。	次回のスプリントについても、短期間として早めの振り返りを実施。		

率・不備・無理 • レビューの段取り • レトロスペクティブの段取り	Sprint 2《7/23–8/5》	Retrospective ページは予め作成の上、リマインドをかけて Demo&Retropective を迎えることでより中身の充実を図る。	予め作成するだけでなく、Retrospective 実施前に記入のリマインドを実施する。		
	Sprint 4《8/21–8/26》	他システム接続のため調整のタイムラグがあった。	事前疎通などまで完了した状態でスプリントに入れると良い。		
	Sprint E《9/30–10/6》	デモにおける時間は限られているため、実装機能についての確認をまず実施したい(デザインの議論は別タイミングとしたい)。	限られた Sprint デモの時間においては実装機能について議論し、デザインの議論は別途タイミングを設けて検討する。	【Keep】 ゴールが達成できているかの検証のために、シナリオを作成しデモを実施。ただし、時間に限りがあるため、デモでは主要シナリオの確認を行い、その後 1 週間の FB 期間を設け、経済産業省での確認を実施。	P1
	Sprint F《10/7–10/13》	デモでの議論で PRD レベルでのディスカッションが増えてきて時間を要している。	現時点での認識や情報を基に、PRD を整理する。	仕様調整資料を基に議論をしていたため、PRD の最新化を実施。	P1

	Sprint G《10/21-10/28》	Day1 でスプリント G の設計レビューをし、feedback でスプリント F の操作を確認し、結合テストでスプリント D、E のシナリオをレビューするなど、タスク毎に異なる機能を同時期に取り扱うため少し混乱した。	3 月向け開発プロセスでは本来やりたかったミニウォーターフォールのプロセスに従い、PRD→DevSpec→Testspec の流れで執筆・レビューができるとその機能についてより深く検討ができる。	3 月向け機能の開発プロセスについて、以下の方針で検討。 ・外部 API、各種手続きは、ウォーターフォール（PRD 作成次第、開発範囲を協議しスケジューリング） ・12 月向け機能改善やユーザテスト要望は、アジャイル（開発期間はバックログのストーリーポイント/ベロシティを基に設定） ・UI/UX 改善は別体制で実施	P1
	Sprint G《10/21-10/28》	細かいですが、本レトロスペクティブのように開催予定日と開催日が異なる場合は、実開催日の記録が欲しい。	ページ内の Date 欄に実開催日を記入する。		
	Sprint G《10/21-10/28》	10/14 のレトロスペクティブの Action に記載がなかった。	今後も Action を記載頂き、次のスプリントのレトロスペクティブの Keep で前スプリントの Action を振り返る。		
会議機材の不足（オンサイトとリモートの混在時）	Sprint D《9/17-9/29》	画面をモニタに投影しながら Slack 上にその画面の共有ができない。	資料投影とモニタ側投影の担当者を分担する。		
	Sprint D《9/17-9/29》	Jabra を使用していても、聞き取りにくい声があった。	発言者以外は Slack のマイクを OFF にして、発言者のみマイクを ON するようにする。		

	Sprint D《9/17-9/29》	参加者がオンラインとオフラインが混在している現状では、オンラインにとって音声が聞き取りにくい、たまに SlackCall に画面が映っていない、ホワイトボードでの議論についていけない、といった問題が起きている。	・集音器を2台とし、机の手前と奥に1台ずつ配置し、発声者以外の人が ON/OFF をコントロールする（機器や運用が厳しければ、発声者が必ず自身の PC のマイクを ON にする運用を徹底する）。 ・Slack 画面に資料などが映ってないことに気づいたら、誰でもすぐに指摘することを徹底する。 ・ホワイトボードでの議論が始まったら、誰かが必ずビデオ共有する。	機材の最適化によって問題はほぼ解消。	
会議の負荷	Sprint D《9/17-9/29》	本ミーティングは長時間(3h 程度)なので、休憩なしはきついと思われる。	Planning の直前に一度休憩時間を設ける。	10 分休憩を取るように習慣付けができている。	

3.2.6. 3 月リリース向け開発終了時（開発全体終了時）

12 月リリース向け開発に引き続いて実施した 3 月リリース向け開発まで終了した時点で実施した振り返りで挙げられた主な「K: To be kept」、「P: Problem 及び T: Try」をそれぞれ以下に示す。なお、挙げられた内容には重複するものもあるが、オリジナルのニュアンスを保つため、あえて統合等の編集は行っていない。

(1)「K: To be kept」

No	区分	To be kept
1	ツール（2.2 項）	• Jira、Confluence、Slack の使い勝手が良く、特に Confluence での情報共有は有効に感じた。
2	ドキュメントプロセス（2.3 項）	• 優先度を大きくは外さなかった。 • これまでのユーザーの声をふまえ大きく外さなかったことは良かった。 • ただ、当初は事務局が J グランツに最も求めているのは「プロセスのカスタマイズができること」だと考えており、実際ユーザーヒアリングや FS の結果でもその課題は大きくクローズアップされていた。 • しかし、ふたをあけると「申請フォームが自由につくれる」ことのほうが、価値は高く受け止められているように思う。 • 今回もし上記に優先度をつけプロセス機能から提供していたら、評価は悪かったと思う。 • やはりプロトタイプの段階から利用者にあてて感触を聞くのが一番確実だと思う。ユーザーがヒアリングで必要だと話す機能と、実際価値が生み出せる機能は違う。
3	ドキュメントプロセス（2.3 項）	• V1 の課題をベースに、V2 ではどの課題を優先的に対応すべきかを整理したことで、やるべきこと、やらなくてよいことを選別することができた。
4	ドキュメントプロセス（2.3 項）	• PRD プロセスは、要求事項を具体的なイメージに落とすのに役立った。
5	ドキュメントプロセス（2.3 項）	• ユーザー課題を軸にソリューションを定義するドキュメント体系 USM、MRD、PRD、DevSpec、Tenet などの活用をしたことは良かった。
6	ドキュメントプロセス（2.3 項）	• オーナーから開発ドキュメントを体系化やフォーマット化、分割化、管理方法を提示したことで、アクセシビリティ、メンテナビリティ、要件/仕様のコミュニケーションにかかるコストをある程度低減できたと思われる（ただし、比較評価はしてない）。

7	ドキュメントプロセス（2.3 項）	• ドキュメントをオープンに共有することにトライした。 • Jira、confluence、miro、backlog など活用し、ドキュメントはオンライン共有できる形、履歴管理するものはチケットにすることをまず考えたのは良かった。
8	Product Backlog プロセス（2.4 項）	• プロジェクト前半のアジャイル、後半のウォーターフォールを通して一貫して「振り返り（レトロスペクティブ）」を継続実施したこと。「常に改善」のモチベーション維持のために有効であった。
9	Product Backlog プロセス（2.4 項）	• PO チームに正しく権限が割り当たり、仕様を決定することができた。
10	Product Backlog プロセス（2.4 項）	• アジャイルにトライした。 • 苦しみながらプロセスを編み出し、試行し、結果 2 回目は回らなかったが、苦戦したノウハウが残ったのは良かった。 • スプリントで日時を明確に区切り、都度デモと振り返りができたのはとても良かった。 • 12 月開発をウォーターフォールでやっていて、仮に受入れまで実機を触っていなかったらリリースはできていなかったと思う。

(2)「P: Problem」及び「T: Try」

No	区分	Problem	Try
1	開発スタイル（2.1 項）	• アジャイル（12 月リリース向け）からウォーターフォール（3 月リリース向け）へ方式を変更した後も、週に複数回の定期的な調整会議を行っており、アジャイルが不向きであったとは必ずしも言えない。 • アウトカム可視化の頻度が下がってしまい、結果的に、成果物の齟齬が多く発生し、諸々のレビューがリリース直前に集中した弊害があった。	• アジャイル、ウォーターフォールいずれの方式を採用する場合でも、2 週間程度のタイムボックス（スプリント等）を決め、MVP（Minimum Viable Product）の必達から逆算したタイムボックスごとの目標・成果物を設定し、事後のレビューと振返りを継続する。
2	開発スタイル（2.1 項）	• アジャイルに適した体制ではなかった ・ツール・管理基準の問題 ・スクラムマスターの不在 ・コミュニケーションの不足 ・アジャイルプロセス重要性の理解不足	• アジャイルを前提とした開発手法、ツール、基準が整備された事業者（チーム）を選定する。 • 同一事業者の場合、教育、プロセス・ツールの試行、チームビルディングを継続する。 • プロジェクトの特性を見据え、どのプロセスが効果的かチームで話し合う。
3	開発スタイル（2.1 項）	• プロダクト価値向上への温度差 ・終始、「この機能がなくても持つか？」という発想だった。 ・ビジョンや目指す姿への共感が足りていなかった。	• プロダクトの目的、Goal、目指すユーザー体験、の共有が進むと意識が変わる可能性あり
4	開発スタイル（2.1 項）	• ビジョンや目的の共有及び教育	※特に具体策の提示なし。
5	開発スタイル（2.1 項）	• 経済産業省側のロールが不鮮明	※特に具体策の提示なし。
6	開発スタイル（2.1 項）	• 進め方についてはベンダーサイドと腹を割って話し合う必要があったように思える	※特に具体策の提示なし。

7	開発スタイル（2.1 項）	• 風通しの良いコミュニケーション環境の不足（当初は開発実施側から物を言い難い雰囲気があった。請負契約が染みついており距離感に困惑。）。	• 可能な範囲で対面での打合せも行い、コミュニケーションしやすい雰囲気を醸成する。 • ざっくばらんなコミュニケーションを気軽に行うため、ラフな資料等を使って気軽に調整を行う。
8	開発スタイル（2.1 項）	• 協業と分業のバランスの問題があった。	• 特にオーナー側で、協業はできていたと思いますが、ロールを分けて分業することも必要。ただし、人数が少なく、他のプロジェクトと兼務しているのが現状なので、バランスが難しい。
9	開発スタイル（2.1 項）	• 設計検討において、ユーザーが置き去りになることが多かった。開発側からは出来ない報告ばかりで、代替案などの提案は検討の場で経済産業省側から出した数も結構多い印象。何のためにシステムを作り提供するのか、開発者本位に寄りすぎていると感じた。	• 根本的に意識を改革する。 ○ （×）出来ないから要件から落とす努力をする ○ （〇）どうすれば実現できるかを考える努力をして提案する
10	開発スタイル（2.1 項）	• 経済産業省側でのレビューコスト（負担）が非常に高かった。	• 最低限のレビューコストはやむなしで、レビュータスクリストでの管理運用まではできた。今後は、重なるレビューのタイミングを調整できると良い。
11	開発スタイル（2.1 項）	• 作業ルールの周知徹底・意識付け不足・不慣れ ○ 機能一覧ベースの開発への変更による手戻り ○ USM に基づく俯瞰が共有できていなかった ○ アウトカムベースではなく Best Effort ベースでの作業となっていた	• プロジェクトメンバーに対する作業ルールの周知徹底、意識付け、情報共有、スキルキャッチアップ等の方法を、プロジェクト計画に含めて検討することとする。

12	開発スタイル（2.1 項）	• 作業ルールが複雑・高負荷（アジャイル及びウォーターフォールのいずれにおいても、様々な調整会・調整ルールが乱立し、調整が非効率となっていた。）	• 5S（整理・整頓・清掃・清潔・躾）、ECRS 改善原則（削減、統合、並べ替え、簡素化）等による点検を、振り返りのアジェンダに加える。
13	ツール（2.2 項）	• 情報管理ツールが複雑・高負荷	• 情報管理に使用するツールを極力絞る。
14	ドキュメントプロセス（2.3 項）、Product Backlog プロセス（2.4 項）	• プロジェクトの進め方や品質管理について、たびたび認識の齟齬が発生していた。	• 各プロセスに係る受発注双方の意識合わせ、発注サイドによるプロジェクト、要求管理、品質管理などこまめな確認が必要。 • 小さなイテレーションによる反復が必要。
15	ドキュメントプロセス（2.3 項）、Product Backlog プロセス（2.4 項）	• レビューを実施するプロジェクト管理体制が不足気味であった。	• 作業実施体制以外に、作業レビュー体制も十分に構築し、体制図・体制表等によって共有する。
16	ドキュメントプロセス（2.3 項）	• オーナー側からプロダクトのビジョンを共有できていなかった。	• プロジェクトの最初に、オーナーとしてどうしたいかの展望をわかりやすくドキュメント化し、プロジェクトを通じて常に全員が参照できるようにする。
17	ドキュメントプロセス（2.3 項）	• UI/UX の改善が後手になった・設計開発に組み込めず、UIUX チームは立ち上げたものの設計開発と別トラックとなってしまった	• 設計段階からデザイナーを入れる。

18	ドキュメントプロセス（2.3 項）	• レビューコストが高かった ・Goal や USM、非機能要件などが共有しきれていなかったと思う。 ・全体俯瞰してユーザー体験を検証できる人が事業者側に少なかった。 ・このため判断がすべて経済産業省に寄せられたことが原因と思われる。	• プロダクトの目的、Goal、目指すユーザー体験、の共有。
19	ドキュメントプロセス（2.3 項）	• 会議で局所的な仕様の議論で結論づけようとすることが多く、他方との整合がずれていった。	• 最初に影響しそうな箇所についての情報を伝える、議論の後に影響がありそうな箇所を別途フィードバックできる期間を設ける。
20	ドキュメントプロセス（2.3 項）	• ドキュメントの所在がまちまち（リンク先からさらにリンク先に飛ばされるなども含む）。	• ドキュメント管理について標準化が必要（Confluence、Backlog、Onedrive、Slack などの利用について目的などを明確化する）。
21	ドキュメントプロセス（2.3 項）	• 想定ペルソナや仮説が少ない。	※特に具体策の提示なし。
22	ドキュメントプロセス（2.3 項）	• 品質基準の不足（UI/UX は設計標準に基づいて進められず手戻り）。	• 使用する品質基準をプロジェクト開始の早期に特定する。 • 品質基準への準拠の確認結果を成果物とする。
23	ドキュメントプロセス（2.3 項）	• 議論するための資料と決定した内容を明示する資料の定義や運用方法が曖昧で、決定した仕様を振り返って確認する際に苦労した。	• PRD、DevSpec に情報を集約する（決定仕様情報の集約）。 • DevSpec のフォーマットを工夫する（決定仕様情報の集約）。 • Confluence の運用ガイドラインを作成して共有する（資料管理）。

24	ドキュメントプロセス（2.3 項）	• アーキテクチャの選定等、基礎的な設計時点で UX に対する考慮が浅かったのではないか。 • ユーザーから挙がる不満は基本的に UX に対してだが、UX より実装のしやすさ等が優先されたように思われる。	• システムに求められる要件を提示するにあたり、なぞそのような要件が生まれたのかの背景も合わせて提供する。
25	Product Backlog プロセス（2.4 項）	• ベンダーと経済産業省の役割を明確にし、意思決定や、方針検討、調整などに経済産業省は重きをおいた方が良い。	※特に具体策の提示なし。
26	Product Backlog プロセス（2.4 項）	• 問題解決やリスク対応のために業務の実施方法、実施環境等を変えるタイミングが遅く、改善効果（変えることのプラス効果）よりブルウィップ効果（変えることのマイナス効果）に繋がる場合があった。	• 業務の実施方法、実施環境等を変える検討においては、改善効果とブルウィップ効果を十分に評価して採否を決めるようにする。
27	Product Backlog プロセス（2.4 項）	• スプリントプロセスの開始までの段取りが急だった。プラットフォームの整備とアプリケーションの基本構造が出来上がってからスタートするべきだったと思う。あるいは、初めのスプリントは前述の基礎部分の構築をタスクにするといった流れのほうが良かったと思う。	• 機能要件に関する開発スプリントの着手可能条件を設ける。 ○ プラットフォームは構築済み ○ アプリケーションとして基礎的な部分が構築済み ▪ ログイン（疑似で可）からポータル、メニュー等への遷移が可能 ▪ メニューから白紙ページへの遷移が可能
28	Product Backlog プロセス（2.4 項）	• 会議で仕様や機能に関する議論が長引くことが頻繁にあった。「この機能って何」「必要だっけ」「多分こういうユースケースがあるかも」などそもそも論に立ち返ることが多く、システムありきの開発に多く見られる傾向に感じた。	• 論点に集中して時間を短縮化するために、課題やゴール、シナリオを参加者全員が理解した上で仕様や機能の議論を行う。

29	Product Backlog プロセス（2.4 項）	• バグトリアージプロセスによって優先度が P2 以下と判断されたものや Won't Fix として扱ったものがほぼ対応から抜け落ちることになった。改善要望に分類されるものはそれでよいとして、不具合として起票されたものはそもそもの仕様を満たしていないのだから、ワークアラウンドの有無に関わらず改修するのが通常の感覚。	• 優先度判断とともに、仕様不備なのか要望なのか不具合なのかといった区分も合わせて対応の要否を判断する。
30	Product Backlog プロセス（2.4 項）	• スプリントのデモの品質が低い。また、受入れの基準も明確ではなかった。	• 受入基準を明確に提示することで、デモに必要な要素を開発者に認識させる。 • デモで求める水準を下げない。
31	Product Backlog プロセス（2.4 項）	• 準備するシステム環境の計画がうまくされていないことで、環境使い回しに苦労する面があった。	• 環境使いまわし計画を、プロジェクトスケジュールの一部として含めて管理することを必須とする。
32	Product Backlog プロセス（2.4 項）	• 開発環境設定の不足・不備がボトルネックとなり、スケジュール遅延につながることがあった。	• 開発するシステムに準ずる粒度で開発環境・稼働環境等の設計、設定等を提示することを要求する。

4. 次年度以降の Action（案）

上記の実証を踏まえ、次年度以降の優先度の高い Action（案）を以下のように絞り込んだ。

No	観点	Action
1	プロジェクト全体	情報管理が適切にできるように、ドキュメント体系を整理する。
2	開発	効果的・効率的な開発が実践できるように、スプリントプロセスの定着を試みる（リソース、スケジュール等の制約に留意した上で）。
3	ユーザー	ユーザー視点に立ったシステムの開発・運用ができるように、ユーザーとのコミュニケーションを向上させる。
4	運用	効果的・効率的な運用が実践できるように、システム監査を行う。