ISACA東京支部調査研究委員会
アジャイルDevOps研究会
2024年2月

アジャイル開発チェックリスト Ver2.0

※1本チェックリストの管理策、具体策は、アジャイル開発の中でも主流となっている「スクラム」を前提とする。
※2文末が「～すること」は必須確認が必要な具体策。但し開発案件に該当しない等、適用しない旨整理することで対象外とすることも可。文末が「～望ましい」は必要に応じて適用。参考情報については、（参考）を文頭に記載。
※3本チェックリストについては、「合意する/合意を得る」は、「関係者から明示的な反対表明がない状態」を指す。例えば何かを決める際に、ある提案に対しスクラムチームのメンバーから異論が出ない状態を指す、など。
※4本チェックリストの中で説明が必要と思われる語句は、(*#)を付し「用語集(統合版)」シート上のNo.に紐づけたうえでまとめて掲載している。

章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
Ⅰ.ITガバナンス			
	1.方向づけ（Direct）		
		1.経営陣は、組織や部門等にアジャイル開発手法を受入れる組織風土や教育を推進し、基本方針を明確にして承認していること。	
			(1)経営陣はアジャイル開発手法の導入にあたり、基本方針を文書化して明確にしたうえで承認し関係者へ周知していること。基本方針は一般的なアジャイル開発手法の原則に従っている必要がある点にも留意すること。 なお、経営陣がアジャイル開発手法を理解していることが手法導入の前提になる例えば、経営陣の中にアジャイル開発の経験者がいる、アジャイル開発手法に関する資格を保持している、又は関連する研修を受講済である等。 (2)経営陣はアジャイル開発手法を前提とした文化・組織風土の形成を中心となり実施していること。例えば、組織全体へ、又はプロジェクトに対しメッセ―ジを発信し手法や手法導入を経営陣自らが推進している。プロジェクト等へ経営陣が積極的に関与している等。 (3)経営陣はアジャイル開発手法の概念や枠組みに関する教育や基本方針が、組織（全社・特定部門いずれもあり得る）、および外部委託先を含むシステム開発プロジェクトの「ステークホルダー」(*1)に周知されるよう推進すると共に執行部門のマネジメントへこれらを指示していること。
		2.経営陣は、開発プロジェクトに対しアジャイル開発手法の適用可否を予め判定し、承認していること。	
			(1)経営陣はアジャイル開発手法による開発プロジェクトを組織として網羅的に把握するための仕組みを整備していること。例えば、開発案件を一覧化して、アジャイル開発手法の案件が特定できること。それらは開発規模や重要性等により、必要に応じて経営陣に報告されている等。 (2)経営陣はアジャイル開発手法の適用可否判定プロセスを設け、規模や重要性等に応じて、経営陣もしくは権限移譲された責任者（可否判断できる経験、スキルがある要員等）が承認していること。
		3.経営陣は、必要に応じてアジャイル開発チームの外にいて開発現場を支援する「リーダー」を選任すること。	
			(1)リーダーは組織のどこでどのようにアジャイル開発を適用するか決めること。 (2)リーダーはその文化を創造し、育み、脈絡を形成する役目を担うこと。 (3)リーダーは作業、作業における課題、プロセス/構造/変革の課題についてオープンであること。 (4)リーダーはアジャイル開発が機能し、チームが経験主義と自己管理を活用して、複雑な問題を解決したり、価値を創出したりすることに対して責任を負うこと。
		4.経営陣は、アジャイル開発手法を適用するプロジェクトについて、すべて、又は一部を外部委託する場合は必要に応じて契約形態がプロジェクトに適しているかを判定し承認していること。	
			(1)経営陣はアジャイル開発手法による開発プロジェクトのすべて、又は一部を外部委託する場合は契約形態の適合性を経営陣もしくは権限移譲された責任者が判定し、必要に応じて経営陣が承認するプロセスを整備して運用していること。 (参考1) 契約形態の例：「IPA：アジャイル開発外部委託モデル契約 (2021.10.6更新)」。(*2) (参考2) 基本契約と個別契約の併用：プロジェクト全体に共通する事項につき、基本契約をまず締結する。その後、小さな単位（機能単位やリリース単位等）ごとに、開発対象と費用がある程度確定したタイミングで個別契約を順次締結する。契約形態は準委任を基本とする。なお、過去には熟達したシステム開発会社が受託した際、緊張感を保つため契約の一部を請負とした実例もある。
		5.経営陣は、アジャイル開発手法に基づくプロジェクトにおけるプロダクト・オーナー(以降、「PO」と表記)を任命して適切な権限及び責任を与えていること。	
			(1)経営陣はPOをプロダクトの成果に責任を持つ役割として、原則1人の個人に割当てること。その人物に要件の優先順位付けや、仕様の決定を行う権限を持たせること。また、POは頻繁に開発者と対話しながらプロダクトの価値を最大化するために作業を行うこと。なお、開発規模や重要性など案件の状況に応じて、経営陣から権限移譲された責任者が任命する場合もある。 (2)経営陣はPOの役割及び責任として以下のスキル等が一般的に求められるためこれらを満たしていると確認すること。 a)POはプロダクトに必要な機能を定義し、その機能を順位付けできるスキルを持っていること。 b)POは開発チームに目的・ゴールを踏まえた機能概要を説明して理解させ、加えてプロダクトビジョンを示すスキルを持っていること。 c)POはプロダクトの業務領域のエキスパートであることが望ましい。 (3)経営陣は大規模開発案件等、複数チームに別れて開発し、各チームにPOを配置する場合など1プロジェクトのPOを複数人で構成する場合は、PO同士の合意形成を充分に行い、PO毎に意見が異なり開発チームが混乱することを予防する対策を講じること。例えば複数のPOの中から最も権限が与えられているPO1人が最終的に意思決定するなど (4)本邦でアジャイル開発を行う場合、多忙なPOをサポートする代理PO(以降、「PPO」と表記)を設置し役割を分担する方法など、組織の実情を踏まえた工夫を凝らすことが望ましい。
	2.モニタリング（Monitor）		
		1.経営陣は、アジャイル開発手法に基づくプロジェクトの進捗状況を関係者から吸い上げて経営報告させ必要に応じてガバナンスレベルの指示を行っていること。	
			(1)経営陣は、経営影響のある問題点や課題をガバナンスレベルまで適宜報告される一連のプロセスを整備し、運用していること。 (2)経営陣は開発規模や重要性に応じて、実際のプロジェクトの進捗や品質・コスト管理等、実態を把握して確認・検証・指示できるプロセス（現場視察を含む）を整備し、運用していること。 (3)アジャイル開発手法を適用するプロジェクトの場合、ユーザーニーズの変化が早いことから、「デイリースクラム」(*3)や「イテレーション(スプリント)」(*4)レビュー等、開発途中の適切なタイミングで情報がマネジメントから経営陣に届き、適宜評価・判断・指示できる仕組みを保持して運用されることが望ましい。
	3.評価（Evaluate）		
		1.経営陣は、アジャイル開発手法に基づくプロジェクトに関する成果を経営レベルで評価する仕組みを保持し、機能させていること。	
			(1)経営陣はプロジェクト終了時の結果だけでアジャイル開発手法に基づくプロジェクトの評価を行わないこと、自社の今後を見据えた評価（プロセスの見直しや改善に繋がる観点等を含む）がなされる仕組みが整備されていること。 (2)経営陣は、アジャイル開発の経験が浅いメンバーを育成すべく経験を積ませる場合は初期の一時的な生産性低下を許容することが望ましい。
		2.経営陣は、アジャイル開発手法に基づくプロジェクトのリスク評価を実施する仕組みを保持し、取締役会等で評価結果を承認していること。	
			(1)経営陣はアジャイル開発手法を活用したプロジェクトを企画、又は開始するにあたり、開発規模や自社に与える影響を踏まえた重要性等に応じて、アジャイル開発手法や該当プロジェクトの特徴を踏まえた経営（ガバナンス）レベルのリスク評価の実施を担当部署に指示すこと。 (2)これらリスク評価結果は、マネジメントレベルや経営陣へ伝達されること。例えば、開発手法をウォータフォールモデル等の従来型から新たにアジャイル開発手法に変更した場合等、要員スキルや品質確保を起因としたビジネス他への影響、柔軟な変更を前提とした専門家（ソフトウェアの会計処理他、会計専門家関与等を含む）の活用やコスト管理等、外部リソースの活用の判定とその影響確認等を行っていること。 (参考)大規模な開発案件では、ウォーターフォール開発とアジャイル開発を組み合わせる(ハイブリッド型開発)ことも選択肢に入る。例えば、要件が明確な部分をウォーターフォール開発で行い、ユーザーインターフェース(UI)など実際のユーザーと意見交換しながら開発する部分はアジャイル開発で行うなど。
	4.監督と監査		
		1.経営陣は、アジャイル開発手法に基づくプロジェクトの可監査性を保証する仕組みを保持して機能させていること。	
			(1)経営陣はアジャイル開発手法を活用したシステム開発プロジェクトにおいてもリスク（システムリスクやプロジェクトリスク、情報セキュリティリスク、サイバーセキュリティリスク等）に応じた適時適切な監査が実施できるよう、監査実施を妨げないルールを策定し運用していること。 (参考) アジャイル開発手法を活用したシステム開発プロジェクトにおいても従来型の開発プロジェクト同様に可監査性が求められる。特に判定・合意の記録化については、電子記録（進捗管理ツールや静止画・動画・録音）や監査部門の各種ミーティングへの参加による確認等、アジャイル開発手法に合わせた記録化のルールを整備して運用することが望ましい。
章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
Ⅱ.企画フェーズ(ITマネジメント)			
	1.計画(Plan)		
		1.案件がアジャイル開発手法と親和性があるか、経営陣またはリーダーは予め判断する基準を定めておくこと。	
			基準を定めるための参考事例2件。 (参考1) 開発対象のチェックポイント) 以下のような場合はアジャイル開発に適していることが多い a) 複雑、不確実で変動対応性が多く、試行しながらの開発が求められる場合 b) エンドユーザの要求や市場環境の変化に伴い、開発対象を柔軟に追加・変更することが求められる場合 c) 継続的なリリースを行い、顧客・市場からのフィードバックを得ながらソフトウェアを改善させる必要がある場合 IPA「アジャイル開発外部委託モデル契約　契約前チェックリスト」>「4. 開発対象のチェックポイント」参照
			(参考2) ステーシーマトリクス＝別図表参照) 「ステーシーマトリクス」はアジャイル開発に適する案件、ウォーターフォールで開発したほうが良い案件を見極める一つの基準を提供している a) 合理的：ユーザーの要求(Requirements)が明確、かつ求められる技術(Technology)も明確な場合はウォーターフォール型開発が適切 b) 政治的：ユーザーの要求が明確でない場合でも、要求される技術が明確であれば、ウォーターフォール型開発が適切な場合がある c) 断定的：新技術を初めて採用するときなど試行錯誤を繰り返す場合、ユーザーの要求が明確であってもアジャイル型開発の方が適切 d) 複雑：開発案件の複雑性が増すにつれ、「プロセスの透明性」、「頻繁な点検」、「振返りと調整」が必要とされ、アジャイル型開発の方が適切 e) 混沌：ユーザーの要求が不透明で、求められる技術も不明という混沌とした状態であれば、開発を推奨しない なお、本章以降の管理策、具体策はアジャイル型開発の中でも主流となっている「スクラム」を前提として進める。
		2.アジャイル開発のノウハウを持つ協力会社へ委託する場合、経営陣またはリーダーはユーザー企業と協力会社双方が契約に基づく権利義務関係について理解し、適切な契約を締結すること。	
			(1)外部に委託する場合でも、ユーザー企業と協力会社双方のプロジェクト関係者がスクラムを始めとするアジャイル型開発について理解していること。 (2)以下の契約モデルを参考に、原則として準委任契約で委託することが望ましい。 （参考）IPA：アジャイル開発版「情報システム・モデル取引・契約書」 a) アジャイル開発外部委託モデル契約（解説付き） b) 契約前チェックリスト c) アジャイル開発外部委託モデル契約書（ひな型） d) アジャイル開発進め方の指針
		3.ステークホルダー(リーダーを含む、以下同様)と「スクラムチーム」(*5)は、「スクラムのルール」(*6)を理解してプロダクトのビジョンを共有し、協働しながら開発を進めることを合意すること。	
			以下の概念、定義は「スクラムガイド2020」、「アジャイル型開発におけるプラクティス活用リファレンスガイド（IPA公表）」に準拠する。 a) ４つのプロセスを包含するプロセスであるスプリント ⇒４つのプロセス：「スプリント計画ミーティング」(*7)、「デイリースクラム」(*3)、「スプリントレビュー」(*8)、「レトロスペクティブ」(*9) b) スクラムの作成物 ⇒「プロダクトバックログ」(*10)、「スプリントバックログ」(*11)、「インクリメント」(*12) c) 合意されたゴール ⇒「プロダクトゴール」(*13)、「スプリントゴール」(*14)、「完成の定義」(*15)

章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
		4.プロダクトオーナー(PO)は、経営陣またはプロダクトの開発規模に応じてリーダーが任命し、スクラムチームが生み出すプロダクトの価値を最大化すること。	
			(1)POは、効果的なプロダクトバックログ管理をすること。 a)POはプロダクトゴールを策定し、スクラムチームとステークホルダーに以下を明示的に伝えること。 ①このプロダクトを使うとできることは何か ②既存のものと何が違うか ③ステークホルダーからの期待は何か ④ステークホルダーにとっての利点は何か b)プロダクトバックログの各項目を優先順位づけした上で詳細化した「プロダクトバックログアイテム」を作成し、スクラムチームとステークホルダーに対し開発するシステムがユーザーにとってどのような価値をもたらすかという「ユーザーストーリー」(*16)を伝えること。 c)プロダクトバックログアイテムは、ビジネスや開発状況に応じて優先順位を変更し適宜並び替えること。 d)プロダクトバックログアイテムにはスクラムチームのみならずステークホルダーにも作業の状態が把握できるよう、見える化され理解されること。 e)a)~d)の作業はPOが行うこともできるが、代理PO(PPO)を置き、担当させることも可。例えば、POはステークホルダーとの調整やリリース判断などに注力、PPOは現行調査、要件取り纏め、プロジェクト管理といったの後方支援を分担するといったやり方などが考えられる。いずれの場合も、最終的な責任はPOが持つこと。 (2)組織（プロジェクトの実施主体）はPOの意思決定を尊重し、プロダクト開発がより円滑に進むよう協力すること。例えば、ステークホルダーがプロダクトバックログを変更したいときはPOと協議し合意を得ること。
	2.構築(Build)		
		1.POはスクラムチームを編成すること。	
			(1)スクラムチームは原則PO1名、SM1名、開発者複数名で構成されること。スクラムチームは機能横断型で、各スプリントで価値を生み出すために必要なすべてのスキルを備えていること。 (2)一度にひとつのプロダクトゴールに集中している専門家が集まった単位であるスクラムチーム内にはサブチームや原則として階層は存在しない。チームとしての決定はメンバー全員の合意を得ること。 (3)スクラムチームは自己管理型であり、誰が何を、いつ、どのように行うかをスクラムチーム内で決定すること。通常、外部からの指示を待つこはない。 (4)スクラムチームは、ステークホルダーと密に連携し、プロダクトに関して必要となり得る研究、実験、開発、検証、運用、保守などすべての活動に合意すること。 (5)POは編成したスクラムチームについて、ステークホルダーの合意を得た旨、記録を残すこと。なお、「合意」とは関係者から異論が出ない状態を指し、「記録」は改ざん防止、日付、関係者氏名の記載を必須とすること。 (参考)十分なスクラム経験がないスクラムチームの場合、スクラムチームメンバーに様々なプラクティスを習得させコミットメントを達成するためにチームが自己管理する「自己組織化」したチームへ成長させるため、コーチによるサポートを受けさせることも検討する。育成のためには一時的な生産性低下を許容することが望ましい。
		2.POは優先順位、ユーザーストーリー、見積り、スプリントの予定で構成されるプロダクトバックログを初回スプリント計画ミーティングまでに作成すること。	
			(1)POは、プロダクトバックログの内容・編集可能性・ユーザーストーリーの優先順位付けに責任を持つこと。 (2)POは、スクラムチームと一緒に「インセプションデッキ」(*17)を基にプロダクトに必要な事柄のをプロダクトバックログに組み込むこと。 (3)POは、ビジネス要求に合わせて、POが許可した者が常にプロダクトバックログに項目追加や修正、詳細の追加、見積り、並び替えし最新化させること。
		3.POはスクラムに十分な経験・実績を有するスクラムマスター(SM)を選任すること。なお、スクラムチームの中でPOはユーザー側、SMは開発者側の代表者という位置づけになる。	
			(1)SMは、スクラムチームとステークホルダーに対し、以下のことを実施することで、スクラムチームとステークホルダーがプロダクトゴールを実現するという責任を果たすこと。 a)スクラムチームとステークホルダーに「スクラムの理論と5つの価値基準」(*18)、プロセス、作成物を全員に理解してもらえるよう支援すること。 b)プロダクトゴール実現化に向けたスクラムチームの有効性を最大化すること。 c)スクラムチームが内部でプラクティスを改善できるようにすること。 d)本邦の組織では、SMの代わりに開発者を統括する「開発リーダー」を置くことも選択肢に入れることが望ましい。
		4.SMは、さまざまな形でスクラムチームを支援すること。スクラムチームのパフォーマンスが最大化するよう各開発者と話し合い、必要と判断すればステークホルダーと調整し、開発者の調達や交代を要請することも含まれる。	
			(1)自己管理型で機能横断型のチームメンバーをコーチすること。 (2)スクラムチームが完成の定義を満たす価値の高いインクリメントの作成に集中し、プロダクトゴールに向けて可能な限り進捗できるようするよう支援すること。 (3)スクラムチームの進捗を妨げる障害物を排除するように働きかけること。 (4)すべてのスクラムイベントが開催され、タイムボックス(*19)の制限が守られるように支援すること。
		5.SMは、さまざまな形でPOを支援すること。	
			(1)プロダクトバックログに従い、スプリント計画ミーティング、リリース計画の立案、リリース計画ミーティング(*20)の日程やテーマを決めること。 (2)効果的なプロダクトゴールの定義とプロダクトバックログ管理の方法を探すことを支援すること。 (3)明確で簡潔なプロダクトバックログアイテムの必要性について、スクラムチームに理解してもらうこと。 (4)複雑な環境においても経験的なプロダクト計画の策定を支援すること。 (5)必要に応じてステークホルダーとの連携を促進すること。 (6)以下の点を考慮しスプリントの長さをスプリント開始前に決め、POと共にリリース計画案を検討すること。 a)プロジェクト期間 b)ステークホルダーのシステム開発への理解度、スクラムチームへの協力、スコープへの理解、ステークホルダー間のコミュニケーション、POとの関係など c)スクラムチームのリスク許容度 d)開発者のスキル、スクラムの経験など
		6 .スクラムチームが大きくなりすぎる場合は、同じプロダクトに専念した、複数のまとまりのあるスクラムチームに再編成すること。	
			(1)複数のスクラムチームは同じプロダクトゴール、プロダクトバックログ、 およびPOを共有すること。 (2)複数スクラムチームをコンロトールするための体制を構築すること。 （参考）以下a)~c)のような少人数(10人以下）のチームを複数作り、各チームが連携する体制であることが望ましい。 a)ステアリングコミッティ ①構成員マネジメントレベルのマネージャー、システムレベルのQA担当者、PO、ビジネスオーナー(各ステークホルダー）、各SM等 ②目的：定期的に(例えば毎週）開催する。システム全体のリリースと進捗の把握、スコープの調整、各チームのミッションの達成度やチーム内状況の把握、各ステークホルダーがすべきリリースの準備に必要な情報の提供、リファインメント(*21)に伴うステークホルダー間の調整をする。 b)スクラムオブスクラム ①構成員：各SM ②目的：毎日開催する。チームごとのデイリースクラム後にデイリースクラムの形式で行う。自チームの昨日の進捗結果と今日の予定、チーム間の影響の調整(依存関係や阻害要因など)、トピックに対する討議の場であり各種調整を行う c)スクラムチーム
	3. 実行(Run)		
		1.開発者は各スプリントにおいて、利用可能なインクリメントを作成するために必要な作業をすべてこなすために全力を尽くすことを確約（コミットメント）すること。	
			(1)開発者が必要とする特定のスキルは、幅広く、作業の領域によって異なる。ただし、開発者は常に次の結果に責任をもつこと。 a)スプリントバックログを作成すること。 b)完成の定義を忠実に守ることにより品質を作り込むこと。 c)スプリントゴールに向けて毎日計画を適応させること。 d)専門家としてお互いに責任をもつこと。 (2)必要な専門知識（事業価値理解、リスク、コンプライアンス等）やソフトスキル（UIデザイン、要件定義、方式設計、等）、技術（ネットワーク、セキュリティ、プロダクト、クラウド等）が揃うよう社内外から開発者を選任すること。 (3)会計専門家等の専門家の関与も必要に応じて予め検討しておくこと。
		2.品質保証部門（規定類をとりまとめる部門）は、スクラムの手法を踏まえた開発規程・ガイドラインを整備して周知の上、開発部門で運用されていること。	
			(1)スクラムに関する開発標準を定めること。社内に規定類をとりまとめる部門がない、あるいは既存の規定類がない場合は、次に示す方法を参考にして標準の代替とし、スクラムチームで合意すること。 a)委託先の協力会社が持つ標準を利用すること。 b)社外の組織（IPA、アジャイルプロセス協議会、エンタープライズアジャイル勉強会、PMI日本支部など）が提供する資料やガイドライン等、あるいはシステム管理基準を参考にすること。 c)標準は以下の内容について定めること。 ①プロセス（スプリント計画ミーティング、リリース計画ミーティング、スプリント、デイリースクラム、スプリントレビュー、レトロスペクティブ等） ②役割（PO(場合によってはPPOも)、SM(場合によっては開発リーダーも)、開発者、コーチ、各ステークホルダー） ③プラクティス（テスト自動化、リファクタリング(*22)、継続的インテグレーション等） (2)ドキュメント化すべき仕様書（例:基本仕様書、アーキテクチャ設計書、運用マニュアル）を明確にすること。ただし、ドキュメントはプロダクトバックログの拡張として位置づけること。 (3)標準的な規約（コーディング規約、仕様書の形式）を整備すること。 (4)プロダクトバックログの構成（優先順位、プロダクト開発要件（ユーザーストーリー）または プロダクト開発要件以外、見積り、スプリント（予定、実績））を定義すること。 (5)現時点でもっとも役に立つツール（開発支援ツールやコミュニケーションツール、CIツール、構成管理ツール等）を最大限に開発者が活用できるよう、組織内のアジャイル開発やツール類の有識者がツール類に関する情報をウォッチし、評価・試用して、その結果を組織（プロジェクトの実施主体）内に共有すること。 (6)(5)のツールを整備した標準的な環境を構築し、各スクラムチームに展開すること。
	4.モニター（Monitor）		
		1.POは経営陣へスクラムチームの状況（進捗、問題、リスク等）について報告すること。	
			(1)スクラムチームの進捗や成果、スプリントで検知した問題やリスクを吸い上げる仕組みが整備され、運用されているか確認すること。
			(2)スプリントで検知した問題やリスクは、優先順位付けされたうえで、重要な問題やリスクは適宜経営陣へ報告されていることを確認すること。
			(3)経営陣へ報告したスプリントで発生した重要な問題やリスクについて、経営陣から対応方針などをフィードバックされていることを確認すること。
			(4)POは開発者に過度な進捗報告を求めず、SMは開発者が業務に専念できる環境を整えること。一方、開発者は日々の作業内容と進捗を公開することで関係者および第三者に対して「見える化」すること。

章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
		2.スクラムチームは監査の受入態勢が整備され、運用されていること。	
			(1)監査実施者とスクラムチームは監査計画立案時に監査対象となるスクラムプロジェクトと事前に監査時期を合意すること。 (2)スクラムチームは、リリース計画、スプリント計画時にどのタイミングで監査が入っても問題が生じないよう、予め想定しておくこと。 (3)監査で要求する証跡が、成果物として存在することを確認すること。 (4)スプリント開始後監査対象となった場合、スプリント計画等適切な時期にそれらを周知し、次回スプリント計画時にどのタイミングで監査が入っても問題が生じないよう、予め想定しておくこと。 （参考）監査で確認すべき観点の例 a) スクラムチーム人選プロセスの「適切性」 b) プロダクトゴール設定プロセスの適切性 c) スプリントゴール設定プロセスの適切性 d) 完成の定義設定プロセスの適切性 e) 各スクラムメンバーの役割とその活動状況 f) 直近のスプリント進捗状況 g) スプリントレビューによる品質評価

章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
Ⅲ.開発フェーズ(スプリント)			
	1. ユーザーストーリー(*16)		
		1.スクラムチーム(*5)は、実現すべき機能を「ユーザーストーリー」として作成した後、それを基に開発者向けに文書化したプロダクトバックログ(*10)を作成すること。	
			(1)「ユーザーストーリー」は、以下4要素から成るユーザーの要望(「エピック」)を基に作成すること。 ① 「誰が」、「どのような」役割で ② 「何を」したい ③ 「なぜなら」どういった価値を生むから ④ そのためには、「何を」必要とする (2)「ユーザーストーリー」は上記「エピック」を以下「INVEST」の特性を持つ、具体的な動作や操作を参考に分解し作成すること。 ① **I**ndependent：独立し、他のユーザーストーリーと重複しない ② **N**egotiable：要求の実現手段を顧客と開発者が交渉できる ③ **V**aluable：ユーザーにとって価値がある ④ **E**stimatable：作業工数見積りが可能である ⑤ **S**mall：見積もりや計算がしやすい小ささに分けられる ⑥ **T**estable：結果をテストで確認できる （参考）ユーザーストーリーを基に、POがプロダクトバックログを作成することになる。ユーザーストーリーは「具体的すぎず、漠然すぎない」よう注意すること。既存システムの改修案件ではユーザーの要望が具体的すぎて交渉の余地がなくなったり、複数部署にまたがる開発案件では要件が抜け落ちることがある。 (3)ユーザーストーリーから以下を分析し、実現すべき機能に優先順位をつけたプロダクトバックログを作成、開発工数/期間を見積もること。 ① 機能要件：利用者(ユーザー)が実現したいこと ② 品質特性：ビジネス価値につながる非機能要件 ③ ビジネス制約：ビジネス価値の足枷になるもの ④ 技術制約：利用者が実現したいことの足枷になるもの (4)プロダクトバックログでは最終的な成果物に至る過程をあまり詳細に記述しないこと。詳細化は各スプリントの開始時に、目標とする機能を備えたインクリメント(*12)を作成する際、「タスク」として記述すること。
	2. スプリントプランニング(計画)(*7)		
		1.スクラムチームは、スプリントを開始する前に当該スプリントで実行する作業の計画を立てスプリントバックログ(*11)を作成し、併せて完成(Done)(*15)の基準も定義すること	
			(1)スプリント計画は、スクラムチーム全体の共同作業によって作成されること。必要であれば、チーム外の専門家や監査・リスク管理部門を含むステークホルダー(*1)をアドバイザーとしてスプリント計画に招待することが望ましい。 (2)スプリント計画にかける時間は、1回のスプリントのサイクルが1か月の場合、最大で 8時間が目安とされ、スプリントサイクルが短ければ、それに応じてプランニングにかける時間も短くすること。なお、タイムボックス(*19)の延長は原則不可、また何らかの理由で 1 つのスプリントを中止する権限はPOにある。 (3)POは、当該スプリントで実現する目標をチームメンバーやステークホルダーに提示すること。この時、単に文書を交付するだけではなく繰返し伝え、実践することで共通認識化されることが望ましい。 (4)スクラムチームは、POが提示した目標をもとに「なぜ(Why)実現する価値があるのか」、および「何を(What)」具体的に実現するか」を明確にし、合意すること。 (5)完成(Done)の基準については、あらかじめ開発規程等、組織内に該当するルールが既に定義されている場合はそれに準拠すること。該当ルールが組織内に存在しない場合は、開発チーム自身が完成(Done)の基準を定義しスクラムチーム内で合意すること。例えば、「スプリントバックログがすべて実装されていること」、「すべてのテストを通過すること」などが挙げられる。 (6)開発者は、プロダクトバックログアイテムをもとに当該スプリントで実現する機能を選択し、スプリントバックログを作成、完成の基準を満たすインクリメントを作成するための作業をスプリント計画として明文化すること。 (7)どのように作業を進めるか(How)は、開発者だけの裁量とすること。また、当該スプリントで手掛けるスプリントバックログアイテムを選択する作業は、開発者が過去の自分たちのパフォーマンス実績や現在の能力を勘案し、1回のスプリント内で完了可能なものとすること。スプリントを繰り返す中で経験を積み、見積もりの精度を上げていくことが望ましい。 (8)スクラムチームは、各スプリントにおいて何が満たされたら当該スプリントにおける成果物を受入れ可能とするかという「受入基準」を設定すること。例えば、「スプリント開始時に計画したユーザーストーリーが実装されていること」、「セキュリティなどの非機能要件も実装されていること」など。ただし最終的に受入可否を判断するのはPOである。 (9)スプリント計画の適切性を後日監査可能とするため、計画が合意に至るプロセスに関する何らかの記録を残すこと。これらの「記録(証跡)」は文書に限らず、写真、録画やツールによる電子記録等も必要に応じて可とする等、スプリント開始前、あるいはプロジェクト開始前にステークホルダーや組織で合意することが望ましい。 (10)本邦ではアジャイル開発も業務委託で行うケースが多いことから、「偽装請負」と見做されるリスクを回避するため、委託先社員から「実施責任者(業務遂行責任者ともいう)」を選任しスクラムチームに入れ、委託元社員とのコミュニケーションは当該実施責任者(業務遂行責任者)を通して行わせること。このようにして、委託先社員が委託元の言いなりではなく「自律性」があることを示すエビデンスを残すこと(後段の各工程でも同様)
	3. スプリントによる開発		
		1.スプリントにより、短いサイクルでインクリメントをユーザーに届けることを繰り返し、最終的にユーザーの要望を満たすプロダクトを完成すること。	
			(1)当該スプリントで実現する目標を達成するために必要なすべての作業は、1 つのスプリント内で完了するよう計画すること。 例えばスクラムの場合、各スプリント中は開発者の自律性に委ねユーザーとの接触は基本的にないことが前提となる。ただし、XP(Extreme Programming)ではユーザー(スクラムのPOに相当する役割の者)がイテレーション中も絶えず開発者と協働するなど、アジャイル開発にもいくつかのバリエーションがある。このように若干の違いはあるものの、短い期間で反復開発(スプリント)し段階的にプロダクトを作り上げていく枠組みは共通である。 またテスト駆動開発(TDD)をスクラムの中に採入れて開発を進めることもあるが、コストとの見合いで実例は少ない模様。 (2)スプリントによって、プロダクトゴール(*13)に対する進捗の点検と適応・調整が一定期間(1週間～1か月)ごとに確実になり、予測可能性を高めること。スプリントの期間を短くすることでより多くの学習サイクルを生み出し、コストや労力のリスクを短い時間枠に収めることが望ましい。 (3)各回のスプリントでは、以下に留意すること。 a) スプリントゴール(*14)の達成を危険にさらす変更はしないこと。 b) インクリメントの品質を低下させないこと(後述のテスト自動化などにより、早期にバグを取り去ること)。 c) スプリント進捗に伴い、プロダクトバックログを必要に応じてリファインメント(*21)すること。 ⇒「7. リファインメント」参照 (4)スプリントバックログの内容の変更は日次ミーティング「デイリースクラム」(*3)で開発者同士共有を行うこと。POや他のステークホルダーが勝手にタスクを追加・変更はしない仕組み、又はスクラムチームもステークホルダーも変更結果を適宜確認できる仕組みとすること。スプリントバックログを変更する場合は(5)項「バーンダウンチャート」のようなツールを駆使して「見える化」することで、開発者に対する過剰な介入を防ぐことが望ましい。 (5)実績値と計画の乖離を一目で把握できる図表「バーンダウン/バーンアップチャート」や、ボトルネックの特定に役立ち作業項目の各種ステータスを表示する図表「累積フロー」などを利用し、スプリントの進捗をスクラムチームのみならず、ステークホルダーからも「見える化」すること。ステークホルダーが普段馴染みのあるツールが他にあれば、それを利用して開発進捗に不安を抱かせない工夫をすることが望ましい。 (6)タスクは1人の開発者が1日で完了できるサイズ以下にしておくこと。また、開発のために必要なタスクの粒度は「1時間単位」まで分解し、属人性を薄めて品質向上を目指すことが望ましい。 (7)開発者は、シンプルなコードによるプログラミング「リファクタリング」(*22)を行うことで、将来的なコードの修正を容易にすることが望ましい。 例えば、 a)コードを変更してもシステムの機能に影響が出ないことをテストによって確認する b)コードのシンプル化は、長過ぎるメソッド、巨大なクラス、多すぎる引数などから、問題と感じる箇所に対して行う、など。 (8)当該スプリントゴールが外部環境の変化などで目指すべき目標が変化した場合、スプリントバックログをリファインメントし次のスプリントに備えること。 (9)スクラムチームに委託先社員がいる場合 ⇒ Ⅲ.開発フェーズ-2.スプリントプランニング(計画)-1.-(10)と同様。
	4. スプリント中のコミュニケーション		
		1.スクラムチームは、チーム内で緊密なコミュニケーションを行うための仕組みを構築すること。	
			(1)スクラムチームは、デイリースクラムなどにより毎日進捗状況をメンバー全員で把握すること。 (2)デイリースクラムなどの定例ミーティングは短いタイムボックス(15分が目安)で済むように共有する情報を絞り、スプリント期間中は原則毎日、同じ時間・場所(オンラインも含む)で開催すること。 (3)開発者は、定められたタイムボックス内で当該スプリントのタスクが消化されるまで上記を徹底すること。 (4)スプリントバックログへのタスクの追加や削除、見積りとその更新は開発者の責任で行い内部で合意すること。なお、内容の変更は定例ミーティングなどの機会を利用してスクラムチーム内で共有すること。 (5)進捗状況に問題が認識された場合、スクラムチームは定例ミーティングとは別に会議体を設定(「ポストモーティム(緊急会議)」)し、改善策を検討する他、スプリントバックログのスコープを調整すること。そのような問題が起こる前に、開発者はデイリースクラム以外にもより詳細な議論をするため頻繁に話し合うことが望ましい。 (6)SMは、課題が簡単に解決できないと判断した場合、ステアリングコミッティーなど上位決定機関に報告し、判断を仰ぐこと。 (7)当該スプリントで実現しなかった機能はスプリントレビュー(*8)ミーティングでステークホルダーと協議し、必要であば次回以降のスプリントで実現させるか等を検討すること。 ⇒「6. スプリントレビュー」参照 (8)デイリースクラムの可監査性を確保するため、作業進捗や問題の所在を後で検証できるよう、何らかの記録(文書、画像、音声など)を残しておくこと。 (9)スクラムチームに委託先社員がいる場合 ⇒ Ⅲ.開発フェーズ-2.スプリントプランニング(計画)-1.-(10)と同様。

章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
	5. CI/CD(継続的インテグレーション/継続的デリバリー)		
		1.スクラムチームは、自動化ツール導入などによりインクリメントの品質を保ち、ソースコードファイルを継続的に統合(インテグレーション)からデリバリー/デプロイまでできる枠組みを作ること。	
			(1)継続的インテグレーション/継続的デリバリーを効率的に行うため、自動化ツールの導入を検討すること。 (参考) a) 自動化ツールは、以下の①～③まで自動的に行う機能を持つことが望ましい。 ① 開発者がコードをリポジトリ(ソースコードやファイルなどを一元的に管理する格納場所)に反映する毎にソースコードを逐次統合(インテグレーション)する。 ② 複数のソースコードファイルを実行可能形式にまとめる(ビルド)。 ③ テストし、アプリケーションを動かす環境に配置する(デプロイ)。 (参考) b) 複数の開発者が並行してコードを作成するため、テストするプログラムはその時点の最新バージョンである必要がある。この点からも自動化ツールを導入することが望ましい。 (参考) c) テスト自動化ツール導入により、プログラムを少しでも修正しリポジトリに入れたら自動的にテストが始まるように設定しておくことが望ましい。 (参考) d) ただし、結合テスト以降の自動化は難しくなることに留意すること。複数のコンポーネントを結合してテストする時間も、定められたスプリント期間内に確保しておくことが望ましい。 (2)継続インテグレーションで失敗したテストが存在する場合はすぐに対処し常にすべてのテストが成功する状態を保つこと。継続的にインテグレーション、ビルド、テストを実施することで問題を早期に発見できる仕組みとなっていることが望ましい。 CI/CD行程では品質低下を生じさせない仕組みが検討され、実装されていること。また実装された仕組みが正しく運用されていることの確認が都度、又は定期的に行われていることが望ましい。 (3)単体で動作するようになったソフトウェアのコンポーネントを組み合わせ、実際に動作する状態に近くして挙動を確認する「結合テスト」以降の自動化のハードルは高くなるが、開発からリリースまで自動化するデプロイメントパイプラインを作り、不良を後続のステージに流さない工夫をすることが望ましい。「ユーザビリティテスト」、ユーザーに対するデモになる「ショーケース」、予めテストケースを設定しない「探索的テスト」など、ビジネス面のテストは自動化できないものがある。
	6. スプリントレビュー(*8)		
		1.スクラムチームは、毎回スプリントの終了時にスプリントレビューミーティングを開催し、プロダクトゴールに対する進捗について話し合うこと。	
			(1)スプリントの長さに応じてレビューのタイムボックスを決めること。なお、スプリント期間が短ければレビューの時間も短くすること。例えば、1スプリントが1か月の場合は4時間、2週間の場合は2時間程度と予め決めておくことが望ましい。 (2)スクラムチームとステークホルダーは、スプリントで何が達成され、何が変化したかについて、確認すること。この情報に基づいて、参加者は次にやるべきことに協力して取組むこと。 (3)スクラムチームはスプリントレビューでステークホルダーに対しプロダクトインクリメントのデモを行い、ステークホルダーからインクリメントに対するフィードバックを受けること。そのためスクラムチームは当該スプリントのタイムボックス内で必要なテストを済ませておくこと。 (4)レビューミーティングで参加者は以下を行うこと。 a) 出来上がったインクリメントがユーザーストーリーに沿っていることの点検。 b) インクリメントが当該スプリントのバックログをすべて消化していることの点検。 c) インクリメントのデモを確認後、当該スプリント終了可否に関する検討。 (5)POは、スプリントレビューの議論と結果を踏まえ、インクリメントの受入れ可否について最終決定し、ステークホルダーが合意した記録を残すこと。 (6)スクラムチームに委託先社員がいる場合 ⇒ Ⅲ.開発フェーズ-2.スプリントプランニング(計画)-1.-(10)と同様。
	7. リファインメント(*21)		
		1.スクラムチームは、スプリントレビューの結果を踏まえ、プロダクトバックログの内容を変更、改めて優先順位付けを見直すリファインメントを行うこと。	
			(1)リファインメントはPOが主催し、SM、開発者との対話を通し内容をレビューすること。 (2)POは、プロダクトバックログに対し詳細の追加、再見積り、優先順位の並び替えや、プロダクトバックログアイテムの受入れ要件(アクセプタンスクライテリア)を洗練させ、改訂を行うこと。 (3)リファインメントがスプリントレビューを踏まえて行われたことが第三者に分かるように、新旧プロダクトバックログ比較などの記録を残すことが望ましい。 (4)こうしたリファインメントによってプロダクトバックログの内容の透明性を高めることが望ましい。 (5)スクラムチームに委託先社員がいる場合 ⇒ Ⅲ.開発フェーズ-2.スプリントプランニング(計画)-1.-(10)と同様。
	8. スプリントレトロスペクティブ(*9)		
		1.スクラムチームは、スプリントレビュー後に別途当該スプリントを振返り次回スプリントにつなげるレトロスペクティブミーティングを開催し、品質と効果を高め開発プロセスを改善する方法を計画すること。	
			(1)レトロスペクティブ(振返り)では、当該スプリントで何がうまくいったか、どのような問題が発生したか、そしてそれらの問題がどのように解決されたか(または解決されなかったか)、どうやったら次のスプリントでよりうまくできるかを議論し、改善策を特定して共有し、次回スプリントに生かすこと (2)1回のスプリントが1か月の場合、レトロスペクティブは最大3時間を目安とすることが望ましい。スプリントの期間が短ければ、それに応じてレトロスペクティブの時間も短くする点に留意すること。 (3)スクラムチームは、個人、相互作用、プロセス、ツール、完成の定義に関して、今回のスプリントがどのように進んだかを点検し、記録を残すこと。 (4)スクラムチームは、終了したスプリントを振り返る議論を通して次回スプリントの有効性改善に役立てること。影響の大きな改善はできるだけ早く対処すべく次のスプリントバックログに追加すること。 (5)レトロスペクティブミーティングを以て1回のスプリントが終了し、間を置かず次のスプリントに向けた計画ミーティングを開くこと(これらを踏まえて「2. スプリントプランニング」に戻り、次のスプリントを開始する)。 (6)スプリントを繰返し、すべてのプロダクトバックログがプロダクトインクリメントの中で実装されステークホルダーに受入れられたとき、当該スクラムは「終了」する仕組みとすること。 (7)1回のスプリントが終了する度に、 a)関係者間のコミュニケーションの円滑化 b)認識齟齬の防止 c)進捗状況の共有 などのために必要なドキュメント類を整備すること。こうしたドキュメント類は、すべてのプロダクトバックログを実装し当該スクラムが終了するまで改訂を続けることから、バージョン管理を徹底すること。 また、運用部門、セキュリティ部門、リスク管理部門、監査部門など、多くの関係者も利用することから、利用者の特性に応じ容易に理解可能な内容・表現とすること。その際、分かり易いソースコードで記述したり、画像を活用したりすることが望ましい。 (8)スクラムチームに委託先社員がいる場合 ⇒ Ⅲ.開発フェーズ-2.スプリントプランニング(計画)-1.-(10)と同様。

章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
Ⅳ.IT運用における継続改善			
	1. IT運用における DevOps のプロセス		
		1. IT運用チームは、アジャイル開発のカルチャを理解し、必要に応じてDevOps手法を運用に取り入れること	
			(1)POは、組織とチームとの間で業務上の整合性を確保していること。 a)リソース、目的、目標、優先度に対する認識が同じ組織内で異なるものにならないよう、業務、開発、運用のチームが認識を共有すること (2)IT運用チームはソフトウェアライフサイクルを確実に理解していること。 a)アプリケーションの全体的なライフサイクルと、そのライフサイクル内での各アプリケーションの段階をチームが理解していること (3)IT運用チームはサイクル時間を減らすことを意識すること。 a)テストの負荷を低く抑えるために、個々のリリースの規模と範囲を制限すること b)ビルド、テスト、構成、デプロイのプロセスを、可能な限り自動化すること c)開発者間や、開発者と運用チームの間のコミュニケーションの妨げとなるものを解消すること (4)IT運用チームはプロセスを見直して改善すること。 a)プロセスと手順に決して完成はなく、絶えず改善することを目的に、現行のワークフロー、手順、ドキュメントに対する定期的な見直しの機会を設けること (5)IT運用チームは、先を見越して計画を立案すること。 a)あらかじめ失敗を織り込んで計画すること b)問題が発生したときにそれを迅速に特定し、対応にあたる正しいチーム メンバーに問題をエスカレートして、解決策を確認するためのプロセスを設けること (6)IT運用チームは、失敗から学ぶこと。 a)運用上の障害が発生した場合、問題をトリアージし、原因と解決策を文書化して、得られた教訓を共有すること b)必要に応じ、今後同じような障害が発生したときに自動的に検出できるように既存のビルドプロセスを更新すること (7)IT運用チームはスピードを最適化し、データを収集すること。 a)できるだけ小規模な変更で作業することを意識すること b)新しいアイデアを試す際には運用データを収集して実験の効果を評価できるように準備し、仮説が間違っていた際にフェイルファスト(*23=早く失敗、安く失敗、賢く失敗)する準備をしておくこと (8)IT運用チームはメンバーの学ぶ時間を計画に折り込むこと。 a)失敗・成功に関わらず、計画時に重要な教訓を蓄積するための時間を見込み、それらの教訓が確実にチームに浸透するようにすること b)チームがスキルの形成、実験、新しいツールや手法の習得を行うための時間を確保すること (9)IT運用チームは操作を文書化しておくこと。 a)製品コードと同じ品質水準で、すべてのツール、プロセス、自動化タスクを文書化すること b)サポートするシステムの現在の設計とアーキテクチャを、復旧プロセスや他のメンテナンス手順と併せて文書化すること c)理論上最適なプロセスではなく、実際に行っている手順に注目し、その文書を定期的に見直して更新すること d)コードに意味のあるコメントが含まれていることを確認すること（特にパブリック API の場合） e)できれば、ツールを使用してコードの文書を自動的に生成すること (10)IT運用チームや他チームと知識を共有すること。 a)ドキュメントを整理し、文書がメンバーから簡単に見つけられるようにすること b)ランチでの歓談（非公式のプレゼンテーション）、ビデオ、ニュースレターなど、工夫して機会を設け知識を共有すること

章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
		2. IT運用チームは、DevOps手法に基づくリリース対応を行うこと	
			(1)IT運用チームは、デプロイを自動化すること。 a)自動化のメリットを理解すること ①より速く信頼性の高いデプロイを実施できる ②テスト、ステージング(*24)、運用など、サポートされている環境への一貫したデプロイが保証される ③人手でデプロイすると発生する可能性があるヒューマンエラーのリスクがなくなる ④都合に応じてリリースをスケジュールしやすくなり、可能性のあるダウンタイムの影響が最小限になる b)テスト、ステージング(用語集参照)、運用環境に各アプリケーションをデプロイするプロセスを自動化すること c)ロールアウト(*25)中の問題をすべて検出するためにシステムを所定の場所に配置し、修正のロールフォワード(*26)や変更のロールバック(*27)のための自動化された方法を用意すること (2)IT運用チームは、継続的インテグレーション(CI)を使用すること。 a)継続的インテグレーション(CI)の特性を理解すること ①一元化されたコードベース(*28)に対し、開発者のすべてのコードを定期的にマージし、標準的なビルドプロセスとテストプロセスを自動的に実行する手法である ②不整合を生じることなくチーム全体が同時にコードベースの作業を行うことができること ③コードの欠陥をできるだけ早く見つけるのにも役立つこと ④できる限り、コードをコミットまたはチェックインするたびにCIプロセスを実行する必要があり、少なくとも1日に1回実行する必要があること b)トランクベースの開発モデル（開発者たちが単一のブランチ(トランク)に対してコミットする仕組み)の導入を検討すること (3)IT運用チームは、継続的デリバリー(CD)の使用を検討すること。 a)継続的配信(CD)の特性を理解すること ①運用環境と同様の環境に対してコードのビルド、テスト、デプロイを自動的に行うことで常時デプロイ可能なコードを確保する手法である ②CD を追加して完全なCI/CDパイプラインを作成すると、コードの欠陥をできるだけ早く検出するのに役立つこと ③適切にテストされた更新プログラムを短時間でリリースできること b)継続的デプロイに必要な堅牢な自動テストと高度なプロセス計画が実現できるかを見定め、適用可否を決定すること (4)IT運用チームは、少しずつ漸進的な変更を行うこと。 a)大規模なコード変更を行うと、小規模な場合よりバグが発生する可能性が高くなるため、 可能な限り変更は小さいものに留めること (5)IT運用チームは、変更の公開を管理すること。 a)更新プログラムがエンドユーザーに提供されるタイミングを確実に管理すること ①フィーチャートグル(*29)を使用しエンドユーザーに対して機能を有効にするタイミングを制御することを検討すること (6)IT運用チームは、リリース管理戦略を導入してデプロイのリスクを軽減すること。 a)アプリケーションの更新プログラムを運用環境にデプロイする一定のリスクを低減するため、カナリアリリース(*30)やブルーグリーンデプロイ(*31)などの戦略を使用し、更新プログラムをユーザーのサブセット(*32)にデプロイすること b)各更新プログラムが想定どおりに動作することを確認した後、残りのシステムに各更新プログラムをロールアウトすること (7)IT運用チームは、すべての変更を文書化すること。 a)軽微な更新や構成の変更が、混乱やバージョン管理の競合を引き起こす原因となる場合があるため、いかに小さな変更でも必ず明確な記録を残すようにする b)適用したパッチ、ポリシーの変更、構成の変更など、変更された点をすべて記録すること c)変更の記録をチーム全員が見られるようにすること ①機密データを含めない。 たとえば、資格情報が更新されたことや、変更したユーザーは記録するが、更新後の資格情報は記録しない。 (8)IT運用チームは、インフラストラクチャをイミュータブル(*33)にすることを検討すること。 a)インフラストラクチャの変更内容を正確に把握するため、運用環境にデプロイした後でインフラストラクチャを変更することを禁止すること b)イミュータブルインフラストラクチャについて理解し、導入可否の検討を行うこと
		3. IT運用チームは、DevOps手法に基づ監視対応を行うこと	
			(1)IT運用チームは、システムを監視可能にすること。 a)IT運用においてシステムやサービスの正常性と状態を常に明確に把握しておく必要があることを理解すること b)外部正常性エンドポイントを設定して状態を監視し、運用メトリックをインストルメント化＝機能の組み込み(*34)するようにアプリケーションをコーディングすること c)状態監視を行う際は、システムの枠を越えてイベントを相互に関連付けるのに役立つ一般的かつ一貫性のあるスキーマ＝監視プラットフォーム(*35)を使用すること (2)IT運用チームは、ログとメトリック(*36)を集計して相互に関連付けること。 a)適切にインストルメント化されたテレメトリシステム(*37&38)からは、生のパフォーマンスデータ(*39)とイベントログ(*40)が大量に生成される。運用スタッフが常にシステムの正常性に関する最新情報を把握できるよう、テレメトリ(用語集参照)とログデータの処理と関連付けがシステムによって短時間で行われることを確認すること b)問題の全体像を把握でき、イベントの相互関係がわかるように、データを整理して表示すること c)データの処理方法とデータの保存期間に関する要件について、会社の保持ポリシーを確認すること (3)IT運用チームは、自動化されたアラートと通知を実装すること。 a)監視ツールを設定して、潜在的な問題や現在の問題を示すパターンや状態を検出すること b)問題に対処できるチームメンバーにアラートを送信すること c)誤検知を避けるためにアラートは継続的に調整すること (4)IT運用チームは、資産とリソースの有効期限を監視すること。 a)証明書など有効期限を有する一部のリソース・資産に対し、有効期限の切れる資産と具体的な期限、その資産に依存しているサービスまたは機能を確実に追跡すること b)資産の有効期限は、自動化されたプロセスを使用して監視すること c)資産の有効期限が切れる前に運用チームに通知し、期限切れによってアプリケーションが損なわれる恐れがある場合は、状況をエスカレートすること
		4. IT運用チームは、DevOps手法に基づ管理対応を行うこと	
			(1)IT運用チームは、運用タスクを自動化すること。 a)反復的な運用プロセスに一定の実行と品質を確保するために、タスクをできる限り自動化すること b)ソース管理を使って、自動化を実装するコードをバージョン管理すること c)他のコードと同様に、自動化ツールをテストすること (2)IT運用チームは、コードとしてのインフラストラクチャのアプローチをプロビジョニング(*41)に採用すること。 a)リソースのプロビジョニングに必要な手動構成を最小限にとどめ、代わりにスクリプトや Azure Resource Manager テンプレートを使用すること b)保守する他のすべてのコードと同様に、スクリプトとテンプレートをソース管理で維持すること (3)IT運用チームは、コンテナ(*42)の使用を検討すること。 a)コンテナの特性を理解した上で、コンテナの使用可否を検討すること ①アプリケーションをデプロイするための標準的なパッケージベースのインターフェイスが用意されている ②コンテナを使用する場合は、アプリケーションの実行に必要なソフトウェア、依存関係、ファイルを含む自己完結型パッケージを使用してアプリケーションをデプロイできる ③アプリケーションとその基になるオペレーティングシステムとの間に抽象化レイヤーを形成し、環境間の一貫性を提供できる ④抽象化レイヤー構造(*43)によって、ホスト上で実行されている他のプロセスやアプリケーションからコンテナを分離することができる (4)IT運用チームは、回復性と自己復旧機能を実装すること。 a)回復性（障害から回復するアプリケーションの能力）について理解し、実装すること ①たとえば一時的なエラーの再試行やセカンダリインスタンス (場合によっては別のリージョン) へのフェールオーバー(*44)がある ②アプリケーションをインストルメント化し、機能停止や他のシステム障害を管理できるよう、問題を即座に報告する (5)IT運用チームは、運用マニュアルを用意すること。 a)運用マニュアルまたは Runbook(*45) の特性を理解すること ①運用スタッフがシステムを保守するために必要な手順や管理情報を文書化したものである ②その他あらゆる運用シナリオが文書化されるほか、サービスの障害など、中断が生じたときに役に立つ軽減計画も文書化される b)これらのドキュメントを開発プロセスの間に作成すること c)これらのドキュメントを作成以降、最新の状態に保つこと d)これらのリソースは、定期的にレビュー、テスト、改善する必要のある生きたドキュメントとして扱うこと e)文書が共有されることがきわめて重要となることを認識し、各自が持つ知識を提供し、共有するようチームメンバーに奨励すること f)チーム全員がドキュメントにアクセスできること g)チームのメンバーのだれもが簡単に文書を更新できること (6)IT運用チームは、オンコール手順を文書化すること。 a)オンコールの職務、スケジュール、手順を確実に文書化し、すべてのチームメンバーと共有すること b)情報を常に最新の状態に保つこと (7)IT運用チームは、サードパーティの依存関係のエスカレーション手順を文書化すること。 a)直接制御できない外部のサードパーティサービスにアプリケーションが依存している場合、機能停止への対応計画を立てること b)計画した軽減プロセス(*46)の文書を作成すること c)文書にはサポートの連絡先とエスカレーションパスを含めること (8)IT運用チームは、構成管理を使用すること。 a)構成の変更を計画し、運用が認識できるようにして、それらを記録すること b)構成管理データベース(*47)またはコードとしての構成アプローチの使用を検討すること c)構成を定期的に監査して、望ましい設定が実際に行われていることを確認すること (9)IT運用チームは、プラットフォームに関するサポートプラン（例えばAzure サポートプラン）を入手し、サポートプロセスを理解すること。 a)ニーズに適したサポートプランを決定し、チーム全体がプランの使用方法を理解するようにすること b)チームのメンバーがプランについて細部まで理解し、サポートプロセスの具体的な流れやサポートチケット申請方法を把握していること c)大きなイベントが予測される場合は、サービス制限の引き上げをサポートセンターに相談すること

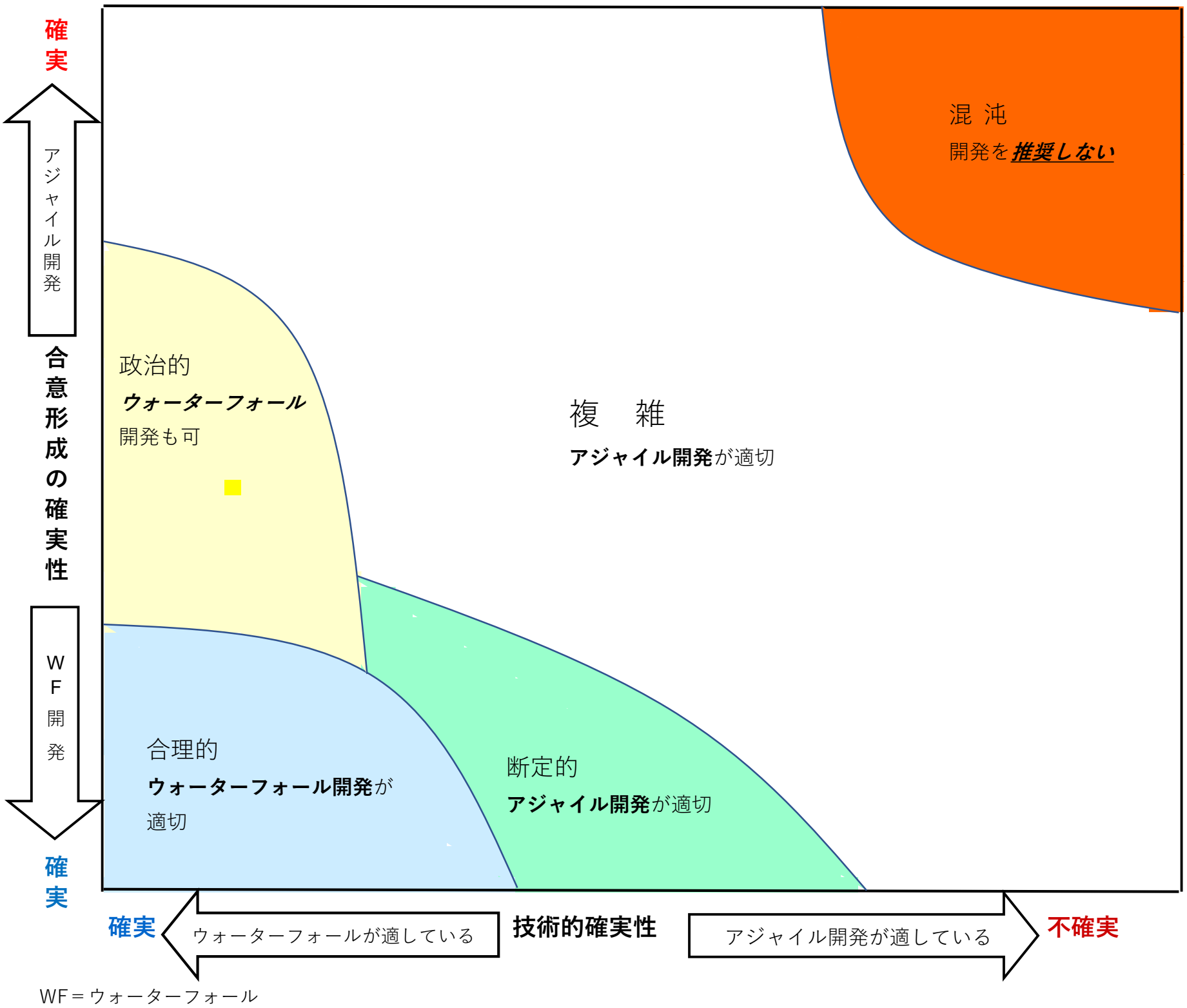
章立て	項立て	管理策	具体策（管理策の補足説明、又は任意選択肢）
			(10)IT運用チームは、リソースへのアクセス権を付与する際に最小限の特権の原則に従うこと。 a)リソースへのアクセス権は慎重に管理すること b)ユーザーにリソースへのアクセス権を明示的に付与しない限り、アクセスを既定で拒否し、タスクの遂行に必要なアクセス権だけを付与すること c)ユーザーのアクセス許可を追跡し、定期的にセキュリティ監査を実施すること (11)IT運用チームは、プラットフォームが準備（例えば Azure ロールベース）したロールベース(*48)のアクセス制御を使用すること。 a)リソースへのユーザーアカウントとアクセス権の割り当てを手動のプロセスで実施することは避け、プラットフォームが準備したロールベースのアクセス制御を使用し、ID・グループに基づくアクセス権を付与すること (12)IT運用チームは、バグ追跡システムを使用して問題を追跡すること。 a)担当者どうしの略式のやり取りでバグの状態を追跡することを避け、バグ追跡ツールを使って問題の詳細を記録し、それに対応するリソースを割り当て、進行状況や状態の監査証跡を残すようにすること (13)IT運用チームは、すべてのリソースを変更管理システムで管理すること。 a)DevOps プロセスのすべての側面を管理およびバージョン管理システムに含めると、変更を簡単に追跡および監査できることを理解すること b)変更管理の対象には、コード、インフラストラクチャ、構成、ドキュメント、スクリプトを含めること c)テスト、ビルド、レビューのプロセス全体を通して、これらすべての種類のリソースをコードとして扱うこと (14)IT運用チームは、チェックリストを使用すること。 a)チェックリストを作成し、継続的にメンテナンスすること b)タスクを自動化してプロセスを合理化する方法がないか、絶えず模索すること
	2. DevOpsを構成する要素		
		1.IT運用チームは、以下の候補を参考にDevOps を構成する要素を選定すること。なお、(6)(7)を除く候補はアジャイル開発と重複しており、同じ手法をアプリケーションにもインフラにも採用することを推奨する。	
			(1)プロジェクト管理 a)Azure Boards等のツールを利用 スクラムやカンバンなどのアジャイル手法は短期間での反復開発が可能であり、DevOpsと相性が良いため (2)Shared version control=コードのバージョン管理 a)Azure 環境では Azure Repos が該当する。分散型バージョン管理システムであり、チーム開発に適している。 (3)コードレビュー a)プルリクエストを用いたコードレビュー プルリクエストを用いることで、コードの品質向上と知識共有が可能になるため (4)Test Automation=テストの自動化 a)Azure 環境では Azure Test Plans が該当する。自動テストにより、コードの品質を維持しながら迅速なリリースが可能になる。 (5)Automated infrastructure=インフラの自動化 a)Azure 環境では Azure pipelines が該当する。 いわゆる CI/CD にあたる。自動ビルド、テスト、デプロイが可能になり、効率的な開発が実現できる。⇒ Ⅲ. 開発フェーズ(イテレーション/スプリント)-5. CI/CD(継続的インテグレーション/継続的デリバリー)、およびⅣ.IT運用における継続改善-3. CI/CD(継続的インテグレーション/継続的デリバリー)参照 (6)Infrastructure as Code=インフラ設定のコード化(*49) a)Azure 環境では、Azure Resource Manager が該当する。インフラコード化により、インフラの変更履歴が管理でき、再現性と柔軟性が向上する。 b)また、Azure Automanage を利用することでインフラ構成管理を実現可能である。 (7)Shared metrics=メトリクスの共有 a)Azure 環境では、Azure の各種メトリクスを Azure Elastic Service で取得・分析・表示することを想定する。システムの状態をリアルタイムで把握し、迅速な対応が可能になる。 (8)ChatOps a)Azure 環境では Azure DevOps や slack が該当する。基本機能であるコミュニケーション機能ではチーム内の情報共有が円滑になり、問題解決が迅速に行える。さらに、ChatOps 機能ではシステムからのメッセージ集約およびメッセージを起点とした自動化の構築ができる。
		2. IT運用チームは、サービス基盤の選定において DevOps（Infrastructure as Code）と相性がよいアーキテクチャを選択すること。例えば、マイクロサービス(*50)を推奨する。	
			(1) Micro Service（マイクロサービス、ビジネス機能ごとにサービスコンテキストを分割し小さく独立的で疎結合な構成、主にコンテナ技術で構成される） a)Azure 環境では Azure Kubernetes Service が該当する。アプリケーションの環境を簡素化し、運用の柔軟性を高める。 (2)マイクロサービスの設計・構築において以下の点を考慮すること。 a)一つのサービスは一つの責任を持つように設計する。サービス間の通信はAPIを使用して行う。 b)各サービスは独立して開発、テスト、デプロイできるようにする。 c)サービスは疎結合にするために必要な情報のみを共有するように設計する。 d)サービスはオーケストレーションツールを使用して管理する。 e)データストアは各サービスが必要なデータのみを保持するように設計する。 f)サービスは自動化されたテストを行い、品質を維持する。 g)マイクロサービスアーキテクチャに適したツールを選定し、導入する。 h)サービスはスケーラブルであることを意識し、負荷に耐えうる設計とする。 i)サービスはセキュリティに配慮し、必要な暗号化、認証、認可を実装する。
	3. CI/CD(継続的インテグレーション/継続的デリバリー) ⇒ Ⅲ. 開発フェーズ(スプリント)-5. CI/CD(継続的インテグレーション/継続的デリバリー)参照		
	4. 監視（モニタリング）		
		1. IT運用チームは、Site Reliability Engineering (SRE) (*51)を活用して提供サービスの信頼性を継続改善するために、提供インフラの運用指標となる SLI/SLO を定め、モニタリングを実施していること。	
			(1)SLI(*52) a)Service Level Indicator の略。「サービスレベル指標」と呼称する、サービスの品質を測る指標。-例えば、サーバーの稼働率、特定プロセスの月間停止時間累計などが該当 (2)SLO(*53) a)Service Level Objective の略。「サービスレベル目標」と呼称する。SLIで計測される値の目標値。・SLI と対になり定義される。例えば SLI がサーバーの稼働率であれば、SLO は「月間の稼働率99.99%以上」のように定義する。 (3)SLA(*54) a)Service Level Agreement の略。「サービスレベル契約」と呼称する。ベンダーと顧客との間で交わされる、提供されるサービスレベルについての合意のこと。・多くの場合はシステムの稼働率（ "MTBF/(MTBF＋MTTR)" で定義される）で可用性を示したものである。複数の SLI/SLO を用いて計算される。

参考文献：

1. 「アジャイル型開発におけるプラクティス活用リファレンスガイド」(IPA ; 2013年3月)
2. 「アジャイルソフトウェア開発宣言の読みとき方」(IPA ; 2018年4月)
3. 「アジャイル開発の進め方」(IPA ; 2018年4月)
4. 「アジャイル開発向けモデル契約案について」(IPA ; 2012年5月 & 2021年10月)
5. 「スクラムガイド」(Scrum.org and Scrum Inc. ; 2016年7月 & 2020年11月)
6. 「アジャイル実践ガイド」(PMI ; 2018年4月)
7. 「ディシプリンド・アジャイル・デリバリ-エンタ-プライズ・アジャイル実践ガイド」(Object Oriented SELECTION)（翔泳社 ; 2013年6月）
8. 「アジャイル開発の本質とスケールアップ変化に強い大規模開発を成功させる14のベストプラクティス」(IT Architects' Archive; 2010年2月)
9. 「SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises」(English Edition ; 2018年5月)
10. 「アジャイルソフトウェア開発向けUML適用ガイドラインVer1.1 」(UMLモデリング推進協議会; 2016年6月)
11. 「アジャイル開発における品質保証」(日本科学技術連盟 ; 公表年不明)
12. 「アジャイルサムライ」(オーム社;2011年7月)
13. 「アジャイル開発への道案内」(近代科学社 ; 2017年9月)
14. 「大規模スクラム」(丸善出版 ; 2019年1月)
15. 「図解で分かるアジャイル・プロジェクトマネジメント」(SCC; 2016年6月)
16. 「アジャイル開発外部委託モデル契約」 (IPA; 2021年10月6日更新)
17. 「チーム改革のスイッチ」(ManasLink; 2020年6月)
18. 「スクラム現場ガイド」(マイナビ;2016年2月)
19. 「情報システム監査実践マニュアル」(NPO法人日本システム監査人協会、第3版 ; 2020年6月)
20. 「品質重視のアジャイル開発」(日科技連;2020年9月)
21. 「SCRUMBOOTCAMP THEBOOK」(日経印刷;2020年5月)
22. 「アジャイル開発のプロジェクトマネジメントと品質マネジメント58のQ＆Aで学ぶ」(日科技連;2020年3月)
23. 「エンタープライズアジャイルの可能性と実現への提言[アンチパターンとその克服事例]」((インプレスR&D; 2019年4月)
24. 「アジャイルソフトウェア開発マネジメント」(富士通FSTユニット; 2020年7月)
25. 「継続的デリバリー ～ 信頼できるソフトウェアリリースのためのビルド・テスト・デプロイメントの自動化」(ドワンゴ; 2017年7月)
26. 「アジャイル開発の教本」(インプレス; 2020年5月)
27. 「誰も教えてくれなかったアジャイル開発」(日経BP ; 2022年4月)
28. 「DevOps チェックリスト - Azure Architecture Center」(Microsoft)(https://learn.microsoft.com/ja-jp/azure/architecture/checklist/dev-ops)
29. 「Azure DevOps Services | Microsoft Azure」(Microsoft)(https://azure.microsoft.com/ja-jp/products/devops)
30. 「SREとは？ DevOpsとの違い」(nifcloud)(https://pfs.nifcloud.com/navi/tech/sre.htm)
31. 「リーダーのためのスクラムガイド」(https://www.servantworks.co.jp/resources/scrum-guide-companion-for-leaders/)

【別図表】

ステーシーマトリクス



WF＝ウォーターフォール

No.	用語	解説	登場箇所	URL
1	ステークホルダー	プロジェクトに関係するすべての人々	Ⅰ.ITガバナンス-1.方向づけ(Direct)-1.-(3) Ⅲ.スプリント-2.スプリントプランニング(計画)-1.-(1)	https://kotobank.jp/word/%E3%82%B9%E3%83%86%E3%83%BC%E3%82%AF%E3%83%9B%E3%83%AB%E3%83%80%E3%83%BC-4934
2	IPA	Information-technology Promotion Agency, Japan 独立行政法人 情報処理推進機構	Ⅰ.ITガバナンス-1.方向づけ(Direct)-4.-(参考1)	IPA： https://www.ipa.go.jp/ モデル契約：https://www.ipa.go.jp/digital/model/ug65p90000001ldr-att/000081484.pdf 契約前チェックリスト：https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Fwww.ipa.go.jp%2Fdigital%2Fmodel%2Fug65p90000001ldr-att%2F000081485.xlsx&wdOrigin=BROWSELINK
3	デイリースクラム	スプリントゴール達成のためにスクラムチームメンバーで行うミーティング	Ⅰ.ITガバナンス-2.モニタリング(Monitor)-1.-(3), (3)-a Ⅲ.スプリント-3.スプリントによる開発-1.-(4)	https://products.sint.co.jp/obpm/blog/daily-scrum
4	イテレーション(スプリント)	一連の工程を短期間で繰り返す開発サイクルのこと。アジャイル開発の一概念。 本チェックリストではアジャイル開発の一形態であるスクラム開発を念頭に原則「スプリント」と表記する	Ⅰ.ITガバナンス-2.モニタリング(Monitor)-1.-(3)	https://it-trend.jp/development_tools/article/32-0022#chapter-2
5	スクラムチーム	スクラム開発において開発を担当するチーム。「プロダクトオーナー(PO)」「スクラムマスター(SM)」「開発者」で構成され、通常5〜9人程度の規模が推奨される	Ⅱ.ITマネジメント-1.計画(Plan)-3 Ⅲ.スプリント-1.ユーザーストーリー-1.	https://appstep.jp/media/scrum-roles-and-tips/
6	スクラムのルール	本チェックリストでは「スクラムガイド2020」を指す	Ⅱ.ITマネジメント-1.計画(Plan)-3	https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf
7	スプリント計画ミーティング	スプリントの最初に開催されるミーティング。スプリントで何を提供しをどのように達成するかをチームが確認する場。	Ⅱ.ITマネジメント-1.計画(Plan)-3.-a) Ⅲ.スプリント-2.スプリントプランニング(計画)	https://www.atlassian.com/ja/agile/scrum/ceremonies
8	スプリントレビュー	スプリントの成果をステークホルダーに披露し、フィードバックを得るイベント。スクラムイベントの最後から2つ目に実施する。	Ⅱ.ITマネジメント-1.計画(Plan)-3.-a) Ⅲ.スプリント-4.スプリント中のコミュニケーション-1.-(7)、6.スプリントレビュー	https://enlyt.co.jp/blog/agile-sprint-review/
9	レトロスペクティブ	スプリントの締めくくりとして、開発チーム内で行われる振り返りの話し合い。	Ⅱ.ITマネジメント-1.計画(Plan)-3.-a) Ⅲ.スプリント-8.スプリントレトロスペクティブ	https://business.adobe.com/jp/blog/basics/agile-retrospectives
10	プロダクトバックログ	プロダクトの実現に必要なものの一覧。ビジネスや開発状況に応じた優先順位の高い順番に並べられる	Ⅱ.ITマネジメント-1.計画(Plan)-3.-b) Ⅲ.スプリント-1.ユーザーストーリー-1.	https://www.publickey1.jp/blog/22/deep_diveregional_scrum_gathering_tokyo_2022.html
11	スプリントバックログ	スクラムチームがスプリント期間中に完成を目指す作業アイテムをリストアップしたもの	Ⅱ.ITマネジメント-1.計画(Plan)-3.-b) Ⅲ.スプリント-2.スプリントプランニング(計画)-1.	https://asana.com/ja/resources/sprint-backlog
12	インクリメント	一回のスプリントで作成した動作するアプリケーション。前回のスプリントからの差分に相当し、インクリメントの総体が最終成果物になる	Ⅱ.ITマネジメント-1.計画(Plan)-3.-b) Ⅲ.スプリント-1.ユーザーストーリー-1.-(4)	https://products.sint.co.jp/obpm/blog/what-is-increment
13	プロダクトゴール	プロダクトの将来の状態を表しており、スクラムチームの長期的な目標	Ⅱ.ITマネジメント-1.計画(Plan)-3.-c) Ⅲ.スプリント-3.スプリントによる開発-1.-(2)	https://www.publickey1.jp/blog/22/_regional_scrum_gathering_tokyo_2022.html
14	スプリントゴール	スプリントの唯一の目的を端的に表した短い文章。スプリントバックログのコミットメント。	Ⅱ.ITマネジメント-1.計画(Plan)-3.-c) Ⅲ.スプリント-3.スプリントによる開発-1.-(3)-a)	https://www.agile-studio.jp/post/apm-sprint-goal
15	完成の定義	成果物が、顧客に使用される準備ができていると判断するためのチェックリスト。 ユーザーストーリーが完了と見なしてよいかの判断に使われる。	Ⅱ.ITマネジメント-1.計画(Plan)-3.-c) Ⅲ.スプリント-2.スプリントプランニング(計画)-1.	https://ssaits.jp/promapedia/glossary/definition-of-done.html
16	ユーザーストーリー	システムがユーザーにとってどのような価値をもたらすのかを示すもの	Ⅱ.ITマネジメント-1.計画(Plan)-4.-(1)-b) Ⅲ.スプリント-1.ユーザーストーリー	https://ssaits.jp/promapedia/method/user-story.html
17	インセプションデッキ	プロジェクトの共通理解を10の質問により明確にし、ステークホルダーと共有するためのツール	Ⅱ.ITマネジメント-2.構築(Build)-2.-(2)	「アジャイルサムライ」(オーム社;2011年7月) https://crowd.itpropartners.com/pieceblog/4158
18	スクラムの理論と5つの価値基準	スクラムに定義された5つの価値基準のこと コミットメント (Commitment) 集中 (Focus) 率直 (Openness) 敬意 (Respect) 勇気 (Courage)	Ⅱ.ITマネジメント-2.構築(Build)-3.-(1)-a)	スクラムガイド2020 https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf
19	タイムボックス	一定の時間枠で、事前に固定されている	Ⅱ.ITマネジメント-2.構築(Build)-4.-(4) Ⅲ.スプリント-2.スプリントプランニング(計画)-1.-(2)	https://asana.com/ja/resources/what-is-timeboxing
20	リリース計画ミーティング	ゴールの定義、ビジョンの概要、案件の優先順位決定後に、いつまでに何を達成する必要があるか、チームメンバー、ステークホルダーの考えを統一する場。	Ⅱ.ITマネジメント-2.構築(Build)-5.-(1)	製品リリース計画を簡素化する5つのステップ - MindManager ブログ https://blog.mindmanager.com/jp/%e8%a3%bd%e5%93%81%e3%83%aa%e3%83%aa%e3%83%bc%e3%82%b9%e8%a8%88%e7%94%bb%e3%82%92%e7%b0%a1%e7%b4%a0%e5%8c%96%e3%81%99%e3%82%8b5%e3%81%a4%e3%81%ae%e3%82%b9%e3%83%86%e3%83%83%e3%83%97/
21	リファインメント	1回のスプリント終了後に行われるレビュー後にプロダクトバックログの内容を変更、改めて優先順位付けの見直す	Ⅱ.ITマネジメント-2.構築(Build)-6.-(2)-a)-② Ⅲ.スプリント-3.スプリントによる開発-1.-(3)-c)、7.リファインメント	https://enlyt.co.jp/blog/agile-sprint-refinement/
22	リファクタリング	繰り返しや冗長なコードを削除し、第三者から理解しやすくすること。	Ⅱ.ITマネジメント-3.実行(Run)-2.-(1)-c)-③ Ⅲ.スプリント-3.スプリントによる開発-1.-(7)	https://it-trend.jp/development_tools/article/32-0046

No.	用語	解説	登場箇所	URL
23	フェイルファスト	「早く失敗、安く失敗、賢く失敗」すること。問題の影響を最小限に抑え、迅速な修正や改善を行うことが可能となる。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-1.-(7)	https://twitter.com/hiroki_daichi/status/1388253385065852928
24	ステージング	ステージング環境のこと。本番公開前にシステムの動作を確認する目的で準備され、本番運用環境にできるだけ近づけた環境を必要とする。	IV.IT運用における継続改善-1.1.IT運用におけるDevOpsプロセス-2.-(1)-a)-②	https://e-words.jp/w/%E3%82%B9%E3%83%86%E3%83%BC%E3%82%B8%E3%83%B3%E3%82%B0%E7%92%B0%E5%A2%83.html
25	ロールアウト	新しいバージョンのソフトウェアを段階的に展開することで、問題を特定しやすくする方法である。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(1)-c)	https://aws.amazon.com/jp/builders-library/ensuring-rollback-safety-during-deployments/
26	ロールフォワード	問題が発生した場合に、以前のバージョンから新しいバージョンに移行することである。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(1)-c)	https://aws.amazon.com/jp/builders-library/ensuring-rollback-safety-during-deployments/
27	ロールバック	問題が発生した場合に、新しいバージョンから以前のバージョンに戻すことである。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(1)-c)	https://aws.amazon.com/jp/builders-library/ensuring-rollback-safety-during-deployments/
28	コードベース	一つのアプリケーションを構成するコード群であり、バージョン管理システム上の同一リポジトリに格納されるものである。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(2)-a)-①	https://xtech.nikkei.com/it/atcl/column/14/110900092/110900003/
29	フィーチャートグル	変更をビルドしてデプロイするアプリケーションに対して、明示的にオンに切り換えない限り変更内容が有効にならない設定を加える手法である。 カナリアリリースの実現方法の一つだが、フィーチャートグルを可能とする機構をアプリケーションに組み込む必要がある点がデメリットと考えられる。フィーチャーフラグとも言われる。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(5)-a)-①	https://circleci.com/ja/blog/canary-vs-blue-green-downtime/
30	カナリアリリース	デプロイ戦略の一種。アプリケーションの新バージョンをリリースする際、まず一部ユーザーのみに限定して公開し、本番環境向けのテストを行ってから一般公開する手法。本番環境で発生する不具合をあらかじめ解消し、より完成度の高い状態で一般公開できるメリットがある。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(6)-a)	https://circleci.com/ja/blog/canary-vs-blue-green-downtime/
31	ブルーグリーンデプロイ	デプロイ戦略の一種。新しいバージョンのアプリケーションを本番環境にデプロイする際、新旧の環境を並行して用意し、切り替えることでダウンタイムを最小限に抑える手法である。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(6)-a)	https://circleci.com/ja/blog/canary-vs-blue-green-downtime/
32	サブセット	デプロイ戦略の一種。アプリケーションの一部機能を限定的に公開する手法。新しい機能を追加する際、全ての機能を一度に公開するのではなく、一部の機能のみを公開してテストを行い、問題がなければ他の機能も公開する。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(6)-a)	https://learn.microsoft.com/ja-jp/azure/devops/pipelines/process/deployment-jobs?view=azure-devops
33	イミュータブル	変更不可のサーバインフラをデプロイのみで提供するアーキテクチャのこと。サーバ個別での変更ができないことで資産管理が明確になり、システムの信頼性とセキュリティ向上、予期せぬ問題や悪意のある変更から保護される。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-2.-(8)	https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/reliability-pillar/rel_tracking_change_management_immutable_infrastructure.html
34	インストルメント化	システムの監視や分析に必要なデータを収集し、可視化することである。システムの状態を把握し問題を早期に発見することが可能となり、システムの可用性やパフォーマンスの向上が期待できる。 具体的には、CPU使用率、メモリ使用量、ネットワークトラフィックなどのデータを収集し、ダッシュボードやグラフで可視化する。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-3.-(1)-b)	https://learn.microsoft.com/ja-jp/azure/well-architected/devops/monitor-instrument
35	スキーマ（監視プラットフォーム）	監視定義を指す。複数のパフォーマンスデータの組み合わせから事象を特定する設定がなされている。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-3.-(1)-c)	https://www.rworks.jp/monitoring/monitoring-column/monitoring-design/25591/
36	メトリック	ITの分野では、対象の属性や状態を一定の基準に基づいて測定したり数値化したものを指すことが多い。例えば、システム管理ツールなどがコンピュータの状態を監視して得られる、CPU使用率やメモリ使用量、ストレージやネットワークのデータ転送量などの測定値のことをメトリックという。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-3.-(2)	https://e-words.jp/w/%E3%83%A1%E3%83%88%E3%83%AA%E3%83%83%E3%82%AF.html
37	テレメトリ	システムやアプリケーションから自動的に収集したパフォーマンスデータを分析した結果をもとに、アプリケーションやインフラストラクチャのパフォーマンスや健康状態を監視し、問題を素早く特定して診断すること、またはそのデータを指す。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-3.-(2)-a)	https://www.logicmonitor.jp/blog/what-is-telemetry
38	テレメトリシステム	テレメトリを収集・分析などで状況を可視化し、事前定義された閾値や条件に基づいてアラートや自動アクションをトリガーするシステムである。テレメトリを使用することで、DevOpsチームはシステムやアプリケーションの動作を把握し、パフォーマンスや信頼性を向上させるためのデータに基づいた意思決定を行うことが可能となる。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-3.-(2)-a)	https://www.logicmonitor.jp/blog/what-is-telemetry
39	パフォーマンスデータ	システムやアプリケーションが動作状況を数値化したデータ。CPU使用率、メモリ使用率、ネットワークトラフィック、エラーレートなどのメトリックが含まれる。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-3.-(2)-a)	https://e-words.jp/w/%E3%83%91%E3%83%95%E3%82%A9%E3%83%BC%E3%83%9E%E3%83%B3%E3%82%B9.html
40	イベントログ	コンピュータシステムに起こった出来事や行われた操作などを時系列に記録したデータである。インシデント発生時の原因調査や特定のメッセージをトリガにしたタスク実行などに用いられる。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-3.-(2)-a)	https://e-words.jp/w/%E3%82%A4%E3%83%99%E3%83%B3%E3%83%88%E3%83%AD%E3%82%B0.html
41	プロビジョニング	設備やサービスに新たな利用申請や需要が生じた際に、資源の割り当てや設定などを行い、利用や運用が可能な状態にすること。対象や分野によって「サーバプロビジョニング」「ユーザープロビジョニング」「サービスプロビジョニング」などがある。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(2)	https://e-words.jp/w/%E3%83%97%E3%83%AD%E3%83%93%E3%82%B8%E3%83%A7%E3%83%8B%E3%83%B3%E3%82%B0.html
42	コンテナ	コンテナ技術によって作成された、ビジネス機能ごとにサービスコンテキストが分割されたものである。マイクロサービスを構成する要素であり、小さく独立的で疎結合な構成であることが特徴となっている。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(3)	https://rookie.levtech.jp/guide/detail/232/
43	抽象化レイヤー構造	情報技術において上位のレイヤーが下位のレイヤーを隠蔽し、上位のレイヤーがより抽象的な概念を扱えるようにする構造である。これにより、上位のレイヤーは下位の詳細な実装に依存せずに、より高レベルの操作や処理を行うことが可能となる。 抽象化レイヤー構造は、ソフトウェア開発やネットワーク設計などの分野で一般的な仮想化として広く使用されている。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(3)-a)-④	https://www.seleqt.net/design/design-system-and-abstraction-layers-a-model-for-better-understanding-and-implementation/
44	フェイルオーバー	稼働中のシステムで問題が生じてシステムやサーバーが停止してしまった際に、自動的に待機システムに切り替える仕組みを指す。HA機能ともいわれ、システムの可用性を高めるための冗長化の一つである。 参考として、フェイルオーバーとは反対に手動でシステム切り替えする方法はスイッチオーバーと呼ばれている。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(4)-a)-①	https://www.idcf.jp/words/failover.html
45	Runbook	元々の語源は一つの業務プログラムの中の多数のプログラムをオペレータが楽に,誤りなく取扱えるようにするために,手順を明確に示した操作指示書のことであるが、本稿ではスクリプト等で記述されオペレータまたは自動処理で機械実行可能としたものを指す。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(5)-a)	https://web-community.jp/ctcs_recruit/about/future.html
46	軽減プロセス	システムの問題を事前に特定し、最小限の影響で解決するためのプロセスである。具体的には、自動化されたテスト、継続的インテグレーション、継続的デプロイメントなどが含まれる。軽減プロセスの実施により、システムの信頼性と安定性が向上することが期待できる。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(7)-b)	https://e-words.jp/w/%E3%83%97%E3%83%AD%E3%82%BB%E3%82%B9.html
47	構成管理データベース	広義にはITSM（ISO/IEC20000）においてITインフラの構成要素を管理するCMDBと言われるデータベースを指す。 ここでは、コードを管理するデータベースを指している。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(8)-b)	https://e-words.jp/w/CMDB.html

No.	用語	解説	登場箇所	URL
48	ロールベース	ロールベースアクセス制御（RBAC）のこと。システム内のユーザーのロールに応じてアクセスとアクションを割り当てることで、同じロールを付与されたすべてのユーザーは同じ権限セットを持ち、異なるロールを付与されたユーザーは異なる権限を持つ仕組みである。	IV.IT運用における継続改善-1.IT運用におけるDevOpsプロセス-4.-(11)	https://www.okta.com/jp/identity-101/what-is-role-based-access-control-rbac/
49	Infrastructure as Code (IaC)	IaC (Infrastructure as Code) は、手動のプロセスではなく、コードを使用してインフラストラクチャの管理とプロビジョニングを行うことを言います。 以下のメリットが挙げられます。 ・毎回同じ環境をプロビジョニングできるようになる。 ・インフラストラクチャの設定がコードをデプロイすることで可能となる。 ・プロビジョニングをテンプレート化でき、自動化を実現できる。	IV.IT運用における継続改善-2.DevOpsを構成する要素-1.-(6)	https://www.redhat.com/ja/topics/automation/what-is-infrastructure-as-code-iac
50	マイクロサービス	ビジネス機能ごとにサービスコンテキストを分割し、小さく独立的で疎結合な構成であることが特徴である。 マイクロサービスはコンテナ技術を採用することで独立したアーキテクチャを採用可能であり、また開発チームがコントロールしやすくなることと相まって DevOps と相性が良い。	IV.IT運用における継続改善-2.DevOpsを構成する要素-2	https://www.intellilink.co.jp/column/agile-devops/2021/072800.aspx
51	Site Reliability Engineering (SRE)	元々Googleが提唱したシステム管理とサービス運用に対するアプローチである。SREの特長は、信頼性をシステムの重要な機能の1つと位置づけている点である。SREでは、サイトやサービスの信頼性を向上させるため、コードによって手作業や繰り返し行われる作業（トイル）を減らしたり、システムを自動化して作業量の増大に対応することを重視している。	IV.IT運用における継続改善-4.監視(モニタリング)-1.	https://pfs.nifcloud.com/navi/tech/sre.htm
52	Service Level Indicator (SLI)	サービスレベル指標。サービスの品質を図る指標(indicator)であり、例えば「サーバーの稼働率」などが該当する。	IV.IT運用における継続改善-4.監視(モニタリング)-1.-(1)	https://pfs.nifcloud.com/navi/tech/sre.htm
53	Service Level Objective (SLO)	サービスレベル目標。SLI で設定した指標に対する目標値である。SLI=サーバーの稼働率の場合、「月間稼働率が99.99%以上」のように設定する。	IV.IT運用における継続改善-4.監視(モニタリング)-1.-(2)	https://pfs.nifcloud.com/navi/tech/sre.htm
54	Service Level Agreement (SLA)	サービスレベル契約。SLO に対する提供者（ベンダー）と利用者（顧客）の間の合意(agreement)を指す。	IV.IT運用における継続改善-4.監視(モニタリング)-1.-(3)	https://pfs.nifcloud.com/navi/tech/sre.htm