付録

表 1 似非アジャイルと理想的なアジャイルの比較

分類	項目	似非アジャイル	理想的なアジャイル
全体像	スプリントの全体像	図 1 に近い.	図 2 に近い.
	開発の流れ	ウォーターフォールモデルの各フェーズを順次実施していく.	開発＆QA を同時に進める.
開発スピード（バグ修正コスト）の要因	単体テスト実施タイミングと頻度	実施しない.	開発と同時に何度も実行する.
	結合テスト実施タイミングと頻度	開発フェーズが終了した後に 1 度実施する.	開発と同時に何度も実行する.
	開発・テスト環境	自身が開発担当するコンポーネントはローカルでデプロイ可能. それ以外の外部コンポーネント（DB 等）はローカルに構築するのが難しく，専用環境のみに有り，各デベロッパーは結合テストをローカルで実施できない.（各デベロッパーは結合テストを行わないか，もし実施するとしても環境が無く難しい）	各コンポーネントはコンテナ技術などで仮想化されており，各デベロッパーの専用環境（ローカル・クラウド上の VM 等）に容易に配置が可能である為，様々なテストが容易に実施可能. SaaSのコンポーネントを利用する場合は，自由に触れる SaaS の開発環境がある.
変更スピードの要因	テストの自動化の割合	ほぼ全て手動実行.	アジャイルテストの 4 章限にあるように，ほとんどのテストが自動化されている. 一部は手動実行.
全般的なスピードの要因	ユーザーストーリーの粒度	巨大で，複数機能を束ねており，1 つのサブシステムレベルになっている. ユーザーの多くの体験をまとめている.	1 つの小さな体験を表現できる程度の機能を含んでいる. 大きくない. 開発対象となるコンポーネントは少なく，基本的に 1 種類である.
	システムアーキテクチャー	複数コンポーネントが密に結合しがちで，独立してデプロイする塊が大きすぎる.	コンポーネントの独立性が高く，開発，テスト，デプロイ，保守，運用がしやすい.
QA の品質	QA 担当者の役割	QA フェーズに入ってから各テストを専門で実施する.	基本的に QA 担当者という概念は存在せず，全デベロッパーが QA を実施する. テストのコーディング＆環境整備を全員が出来る. QA 担当者を用意する場合は，探索的テストを実施したり，全体的なテスト環境整備を主導する役割を担うパターンも考えられる.
市場のビジネスニーズへの追従の成功要因	ユーザーストーリー作成プロセス	企画部門の指示内容を要件定義する形で，スプリントを始める前に一気に作る.	毎スプリントのリリース後にフィードバックを得て / リリース物に仕込んだフィードバック機能でフィードバックを得て 新規ストーリーを作り続ける.
	ユーザーストーリー作成者	プロダクトオーナーのみ.	アジャイルチーム内の全デベロッパーとプロダクトオーナー
	ユーザーストーリーの優先度づけの仕方	明確な基準はなく，システム全体を作るために必要な機能を全て実施していく.	ビジネス価値を主眼におき，率先してフィードバックを貰いたいコア機能を優先する.

	リリース後のプロダクト改善	重大なバグは対応する.	リリース後のフィードバックに基づき，新しいユーザーストーリーを作成し，改善し続ける.
	リリース後の体制	開発チームは解散している事がある / 別の開発に着手している.	開発チームが残り，プロダクトの改善を続ける.
チームのパフォーマンス計測の指標	ユーザーストーリーの完成にかかる期間	1スプリントで終わらない.	スプリント内に完了する.
	リリース頻度	年に1回程度	数週間（スプリント終了毎）に1回程度

表 2 似非アジャイルと中間解アジャイル、理想的なアジャイルの比較

分類		項目	似非アジャイル	中間解アジャイル	理想的なアジャイル
全体像		スプリントの全体像	図1に近い.	図3に近い.	図2に近い.
		開発の流れ	ウォーターフォールモデルの各フェーズを順次実施していく.	開発（設計とコーディング）と単体テストを同時に進める. それ以降のテストは順次実行する.	開発＆QA を同時に進める.
開発スピード（バグ修正コスト）の要因		単体テスト実施タイミングと頻度	実施しない.	開発と同時に何度も実行する.	開発と同時に何度も実行する.
		結合テスト実施タイミングと頻度	開発フェーズが終了した後に1度実施する.	Mock & Stub を利用したテストは開発と同時に何度も実行する. 実際の結合対象コンポーネントとのテストは開発フェーズが終了した後に実施する.	開発と同時に何度も実行する.
		開発・テスト環境	自身が開発担当するコンポーネントはローカルでデプロイ可能. それ以外の外部コンポーネント（DB等）はローカルに構築するのが難しく，専用環境のみに有り，各デベロッパーは結合テストをローカルで実施できない.（各デベロッパーは結合テストを行わないか，もしやるとしても環境が無く難しい）	自身が開発担当するコンポーネントはローカルでデプロイ可能. それ以外の外部コンポーネント（DB等）はローカルに構築するのが難しく，専用環境のみにデプロイされており，各デベロッパーは実際のコンポーネントとの結合テストをローカルでは実施できない.	各コンポーネントはコンテナ技術などで仮想化されており，各デベロッパーの専用環境（ローカル・クラウド上の VM 等）に容易に配置が可能である為，様々なテストが容易に実施可能. SaaS のコンポーネントを利用する場合は，自由に触れる SaaS の開発環境がある.
変更スピードの要因		テストの自動化の割合	ほぼ全て手動実行.	単体テスト，一部のコードレビュー（Lint等），Mock & Stub を利用した結合テストは自動実行. それ以外のテストは手動実行.	アジャイルテストの4章限にあるように，ほとんどのテストが自動化されている. 一部は手動実行.

第３研究コース（ソフトウェアテストグループ）

全般的なスピードの要因		ユーザーストーリーの粒度	複数コンポーネントが密に結合しがちで，独立してデプロイする塊が大きすぎる．	1 つの小さな体験を表現できる程度の機能を含んでいる．大きくない． 開発対象となるコンポーネントは少なく，基本的に 1 種類である．	1 つの小さな体験を表現できる程度の機能を含んでいる．大きくない． 開発対象となるコンポーネントは少なく，基本的に 1 種類である．
		システムアーキテクチャー	巨大で，複数機能を束ねており，1 つのサブシステムレベルになっている．ユーザーの多くの体験をまとめている．	コンポーネントの独立性が比較的高い． サイズの小さいユーザーストーリーを各デベロッパーが 1 つずつ並行して開発する際，互いの開発対象コンポーネントが被らない/被る領域が少ないため，コード管理システムでコンフリクトが発生しづらく，開発がしやすい． また，コンポーネントがあまり複雑に結合していないため，テストも実施しやすい．	コンポーネントの独立性が高く，開発，テスト，デプロイ，保守，運用がしやすい．
QA の品質		QA 担当者の役割	QA フェーズに入ってから各テストを専門で実施する．	開発からチームに入り，自動テストなどの環境整備や指導を行い，全デベロッパーが単体テスト，Mock & Stub を用いた結合テストをコーディングできるように教育する．開発フェーズ以降のテストフェーズを主担当で実施する．	基本的に QA 担当者という概念は存在せず，全デベロッパーが QA を実施する．テストのコーディング＆環境整備を全員が出来る．QA 担当者を用意する場合は，探索的テストを実施したり，全体的なテスト環境整備を主導する役割を担うパターンも考えられる．
市場のビジネスニーズへの追従の成功要因		ユーザーストーリー作成プロセス	企画部門の指示内容を要件定義する形で，スプリントを始める前に一気に作る．	リリース後，偶にユーザーフィードバック取得し，それを元に新規ストーリーを作り続ける．	毎スプリントのリリース後にフィードバックを得て / リリース物に仕込んだフィードバック機能でフィードバックを得て 新規ストーリーを作り続ける．
		ユーザーストーリー作成者	プロダクトオーナーのみ．	アジャイルチーム内の全デベロッパーとプロダクトオーナー．	アジャイルチーム内の全デベロッパーとプロダクトオーナー．
		ユーザーストーリーの優先度づけの仕方	明確な基準はなく，システム全体を作るために必要な機能を全て実施していく．	ビジネス価値を主眼におき，率先してフィードバックを貰いたいコア機能を優先する．	ビジネス価値を主眼におき，率先してフィードバックを貰いたいコア機能を優先する．

		リリース後のプロダクト改善	重大なバグは対応する.	リリース後のフィードバックに基づき，新しいユーザーストーリーを作成し，改善し続ける.	リリース後のフィードバックに基づき，新しいユーザーストーリーを作成し，改善し続ける.
		リリース後の体制	開発チームは解散している事がある / 別の開発に着手している.	開発チームが残り，プロダクトの改善を続ける.	開発チームが残り，プロダクトの改善を続ける.
チームのパフォーマンス計測の指標		ユーザーストーリーの完成にかかる期間	1 スプリントで終わらない.	1 スプリントで終わる事もあれば，2 スプリントで終わる事もある.	スプリント内に完了する.
		リリース頻度	年に 1 回程度	数ヶ月に 1 回程度	数週間（スプリント終了毎）に 1 回程度

表 3 比較表の各項目と，対応する問題・分類の対応表

表 1，2 の項目	対応している問題・分類
スプリントの全体像	全体像
開発の流れ	アジャイルのコンセプトの理解不足
単体テスト実施タイミングと頻度	アジャイルのコンセプトの理解不足 個人の技術不足
結合テスト実施タイミングと頻度	アジャイルのコンセプトの理解不足 個人の技術不足
開発・テスト環境	個人の技術不足
テストの自動化の割合	個人の技術不足
ユーザーストーリーの粒度	アジャイルのコンセプトの理解不足
システムアーキテクチャー	個人の技術不足
QA 担当者の役割	アジャイルのコンセプトの理解不足
ユーザーストーリー作成プロセス	アジャイルのコンセプトの理解不足
ユーザーストーリー作成者	アジャイルのコンセプトの理解不足
ユーザーストーリーの優先度づけの仕方	アジャイルのコンセプトの理解不足
リリース後のプロダクト改善	アジャイルのコンセプトの理解不足
リリース後の体制	アジャイルのコンセプトの理解不足 組織や社内ルールに起因する問題 *中間解では出来るだけ回避している「組織や社内ルールに起因する問題」ではあるが，アジャイルの根本的な前提である，リリース後の改善に必須となる条件なので記載している．また，アジャイルのコンセプトの理解不足を解消する為にも記載している.
ユーザーストーリーの完成にかかる期間	各項目の実践レベルによって変化するメトリクス
リリース頻度	各項目の実践レベルによって変化するメトリクス