特集論文

パブリッククラウド上のサービス提供システム構築でのアジャイル開発手法の実践

吉川晃平* 山口能一*
屋敷孝典*
北出晋一*

Practices of Agile Development Method in Construction of Service on Public Cloud

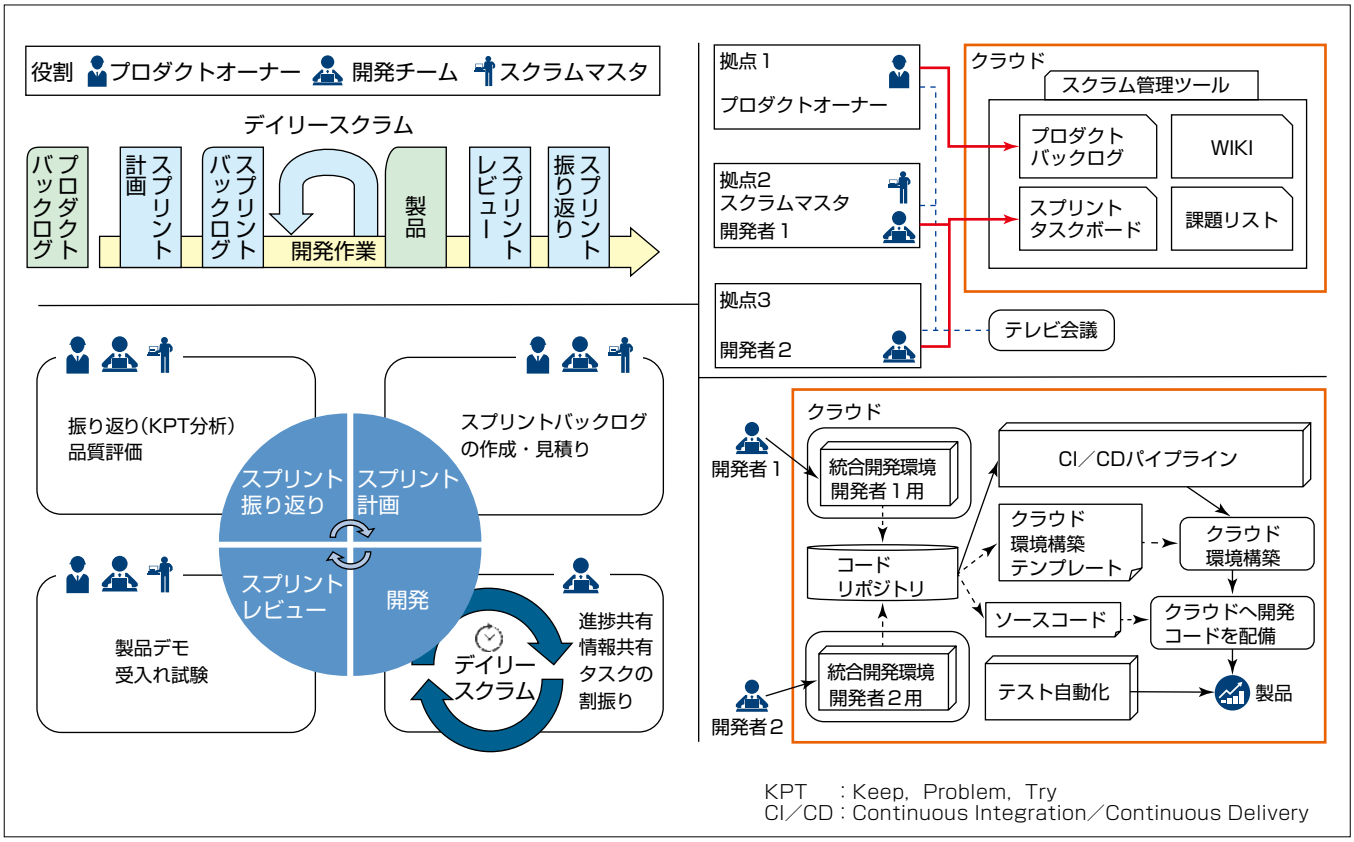
Kohei Yoshikawa, Takanori Yashiki, Shinichi Kitade, Yoshikazu Yamaguchi

要　旨

パブリッククラウド(クラウド)上でのサービス提供システム構築では，クラウドの仕様が頻繁に追加・変更される。変化へ対応しつつサービスを開発するため，今改めてアジャイル開発が注目されている。アジャイル開発の一手法であるスクラムは，開発チーム間のコミュニケーションを重視して開発を進める手法である。スクラムでは顧客を含めた開発チームで編成し，“プロダクトバックログ”へ積み上げた製品要件を優先度順に製品へ実装する。“スプリント”と呼ぶ短い開発期間の中で，計画・設計・コーディング・テスト・受入れ・振り返りを行い，動作可能な製品を評価し，要件の変化を受け入れつつリリースを目指した開発を行う。

スクラム開発を計画可能な状態にするには，スプリントごとの生産量“ベロシティ”の安定化が重要である。そのためには，バックログに対する見積りと生産実績の相対的な計測によって，チームが生産性経験を見積りにフィードバックできるように推進する必要がある。また，短い期間で反復しながら行われる開発では，支援ツール利用によるアジャイル開発の生産性向上，及びクラウドが提供する開発サービスの活用が重要である。

なお，アジャイル開発には多くの利点がある一方，従来型開発と異なる課題に直面することがある。三菱電機インフォメーションシステムズ㈱(MDIS)は顧客と協調し，アジャイル開発の経験を積み上げ，最新の技術を活用したより良いサービスを提供できるよう取り組んでいる。



スクラム開発の流れとクラウド環境での分散開発環境

左上の図は，スクラム開発での役割と開発の流れを示す。左下の図は，一回の開発サイクル(スプリント)に行う作業の流れを示す。右上の図は，クラウド上のスクラム管理ツールを用いた分散拠点でのスクラム開発を示す。右下の図は，クラウド上のサービスを活用した開発環境の構成を示す。

*三菱電機インフォメーションシステムズ㈱

1. まえがき

近年，IT技術の高度化や開発サイクルの短期化，サービスの複雑化などの要因によって，ウォーターフォール型開発が適さないケースが増えている。中でも，クラウド上でのシステム構築では，新サービスのリリース，仕様変更などが度々発生して要件を確定させることが困難な場合がある。アジャイル開発では，短期間の開発サイクルを繰り返してシステムを構築することで，要件の変化を吸収しながら開発することが可能である。

本稿ではクラウド導入案件でアジャイル開発手法を実践的に取り組んだ際の手法，効果，課題を述べる。

2. クラウド上のサービス提供システム構築での課題とアジャイル開発採用の意義

2.1 クラウド上のシステム構築での課題

クラウドは多数の機能をサービスとして提供し，また短い期間で新サービスのリリースや既存サービスの仕様変更を行う。そのため，システム開発の現場ではクラウドが提供するサービス仕様に対して顧客の期待との差異が生じる。時には要件が曖昧なまま“クラウドならできる”との期待から開発企画を始め，後にトラブルにつながるケースが発生し得る。開発者も矢継ぎ早にリリースされるクラウドサービスの全てを把握することは困難であるため，対応が後手に回りやすい。この傾向は生産性を高めるためにクラウド特有の先進的サービスを活用したシステムほど直面しやすい。ウォーターフォール型開発ではフェーズごとに顧客と開発者が仕様を固めることで品質を確保する前提である。しかし，先に述べた理由によって実装前に固めた設計がクラウドへの実装フェーズで覆ると，プロジェクト工程の終盤で取り返しのつかない影響の大きな課題が発生するリスクを生みやすい。

2.2 アジャイル開発採用の意義

アジャイル開発[1]は，包括的なドキュメントよりも動くソフトウェアを，計画に従うことよりも変化への対応を価値とする。これによって早い段階でクラウドに対する実装上の課題を把握して仕様変更を含む対策を顧客と協議し，顧客の持つ暗黙の要件が早期に把握可能になる。開発進捗についても，数値的な進捗度では表しにくい製品そのものの完成度をその都度顧客が把握できる。またプロセスやツールよりも個人との対話を，契約交渉よりも顧客との協調を重んじ，顧客と開発者はそれぞれの役割に従った比較的平等な立場で議論を行い，より良い製品を早期にリリースするために尽力する。顧客はプロジェクトの状況と品質を把握・制御可能な状態と感じ，満足したプロジェクト運営を行うことができる。

3. スクラムによるアジャイル開発の流れ

アジャイルの開発手法として国内で採用事例が多いのがスクラムである。次にスクラムでの開発の流れを述べる。

3.1 アジャイル開発の導入準備とスクラム

スクラムでは，スプリントごとに定めた要件“バックログ”を元に動作可能な製品をリリースする。スプリント期間は１～２週間程度に固定する。体制は顧客を含めたチームとしてスクラム上の役割を定義し，“プロダクトオーナー”が製品要件に責任を持ち，“スクラムマスタ”がスクラム運営の推進支援を行う。“開発チーム”がシステム開発を行い，スプリントの中でバックログを作業に分解した“タスク”を用いて日々開発を推進する（**図１**）。

3.2 プロダクトバックログの検討

プロダクトバックログは，スクラムでの製品に対する要件のリストである。プロダクトオーナーは製品のニーズを分析し，システムが処理すべき機能の洗い出しとユーザーストーリーの立案を行う。そしてリリース計画を立て，優先度の高いバックログから要件を詳細化して受入れ試験可能な状態にする。

3.3 スプリントの計画

スプリント開始時に全メンバーで計画を立てる。まずスプリントで実装する要件“スプリントバックログ”をプロダクトバックログから優先度順に選択し，開発チームが見積りを行い，スプリント内に実現する総生産量をプロダクトオーナーと合意する。その後，開発チームはスプリントバックログを実装に至る実作業（タスク）に細分化する。これには設計，試験計画，ドキュメント作成，受入れデモ準備を含む。

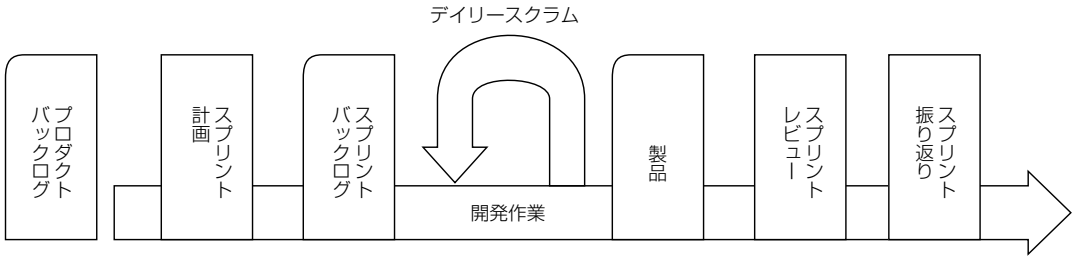


図１．スクラム開発での役割と開発の流れ

3.4 スプリントの実施

開発チームは“デイリースクラム”と呼ぶ会議を毎日実施し(15分程度)，日々担当者が何をすべきか，問題点は何かを明確にして作業を開始する。タスクの分担は機能ごとに専任を設けず開発チームが有機的に互いの作業を支援しあえるように努力する。

3.5 スプリントの成果確認方法

成果報告は受入れ試験と兼ね，デモ形式で実施してプロダクトオーナーによる合否判定を受ける。報告資料はスプリントバックログごとに詳細に記載し，共通的な資料の変更が発生する場合は併せて報告する。デモ手順はバリエーションにこだわらずユーザーストーリーに基づく明確な完了条件を記載することで，合否判定を分かりやすくしたものにする。

3.6 スプリントの振り返り

スプリントの振り返りでは，ベロシティを報告し，KPT分析による振り返りと改善を行う。問題点を早期発見，早期解決することによってスプリントをうまく回すことができる。

4. 開発実践での課題と工夫

アジャイル開発の実践ではウォーターフォール型開発とは異なる課題や非効率性が発生する。MDISはクラウド上のサービス提供システム構築にアジャイル開発を採用する際，以下の点を考慮した開発推進を行っている。

4.1 品質マネジメント対応とインセプションデッキを用いた開発計画の立案

開発標準プロセスの多くはフェーズごとに文書による完了条件を定めた品質確保を前提にしている。一方スクラム開発では明確なフェーズの分離がない。品質マネジメント基準をクリアするために，成果文書の開発標準プロセスへのマッピングを行う事例が三菱電機グループ内で紹介されている。しかしながら，スクラムを初めて導入する開発チームには準備すべき計画項目が多岐にわたりすぎて，全体把握を困難にする場合がある。スクラム導入時は，ウォーターフォール型開発に慣れ親しんだメンバーへのスクラム流儀の意識づけが重要である。その際は，プロダクトオーナーとスクラムマスタが中心となり，アジャイル開発での簡易なプロジェクト憲章である“インセプションデッキ”[2]を作成してプロジェクトのミッション，状況，ビジョン，ステークホルダーの役割等をすみやかにチームで共有させる。

4.2 スプリントバックログに対する見積り精度の改善と計測

スクラムではスプリントバックログに対する生産規模の見積りを，開発チームが合議で過去の実績を基にした相対的な見積値で決めることを原則とする。スクラム開始後しばらくは，開発チームによる開発実績がないため相対見積りはメンバー個人の経験に依存したバラバラな値となりがちである。そのため，合議的な見積りにメンバーも自信が持てず，従来型の工数積み上げ型見積りや識者にゆだねた見積りを望む声が上がる。しかし，アジャイル開発の生産力の源泉は開発チームの主体性と自己責任にあることをスクラムマスタは信じ，メンバーとともに合議的な相対見積りを推進すべきである。同じメンバーでスプリントを重ねていくことで，見積りの元となる生産実績が増えるとともに開発チームはチームとしての生産力を計測できるようになる。その結果スプリント計画時の見積りがそろい始め，またベロシティが計画値に近づいて安定していく。これまでの実績では3サイクル目ぐらいからプロダクトバックログとストーリーポイントの相対値が分かってくるようになる。

4.3 稼働負荷の安定化

開発チームは業務外作業・休暇等開発に割けない時間が発生することが常である。そのため，スプリントバックログごとの見積りポイント達成状況以外に，スクラムマスタはスプリント期間中の総稼働時間を測定し，開発チームが十分開発に安定して取り組めたかを評価するようにする。これによってベロシティの計画・実績の剥離が見積りのぶれによるものか，稼働時間確保の不安定性によるものかを振り返る。稼働時間確保が不安定な場合は顧客・開発チームと改善策を協議する。

4.4 ツール利用によるアジャイル開発の生産性向上

アジャイルソフトウェア開発宣言では“プロセスやツールよりも個人との対話に価値を置く”と謳(うた)われているが，生産性を上げるために適切なツールの利用は不可欠である。MDISが取り組んだ開発では，次のような生産性を阻害する課題に対して，ツールを利用することで解決を図った(**図2**)。

(1) メンバーの地理的分散

地理的問題によって開発チームが同一場所にいない。本来，スクラム開発では開発チームが地理的に一か所で開発を進めるのが良い。しかし，製造業では地方に工場や開発拠点を持つことが多く，また複数社でチームを構成するケースがある。

(2) 進捗・担当管理

スプリントバックログの進捗管理が属人的となりがちであり，ツールで進捗状況を客観的に把握する必要性がある。

4.4.1 プロジェクト管理ツールの活用

アジャイル開発でもプロジェクト管理ツールの導入は有効である。ツールの選定は案件の特性を考慮して行う。

オープンソースソフトウェア(OSS)プロジェクト管理ツール“Redmine”や商用プロジェクト管理ツール“Backlog”[注1]は汎用のプロジェクト管理ツールで，タスク単体

の情報管理に優れ，チケットをガントチャートやカレンダー，ロードマップなどで表示ができるほか，構成管理ツールとチケットを連携させることも可能である。ただし，一方でスクラムでは頻繁にバックログに紐(ひも)づくタスク・担当・優先度・見積値を見直すことがあり，またスプリント中は様々なタスクを開発者が日々共有するため，汎用プロジェクト管理ツールのチケット管理機能はその用途として使いやすいとは言えない。

一方，アジャイル向けのプロジェクト管理を目的としたツールもある。OSSツール“Taiga”はチケット管理やドキュメント管理の機能はシンプルであるが，スクラム手法とカンバン手法のタスク管理に特化しており，スプリントバックログのリストアップ・並べ替え・スプリントへの割当て・スプリントごとのベロシティ計算といったスクラム開発で頻繁に行う煩雑な作業を効率的に行える。またタスクボード機能は，スプリントバックログごとにタスクのグループを生成し，詳細化したタスクとバックログの関連を管理しやすくしている。

(注1) Backlogは，㈱ヌーラボの登録商標である。

4.4.2 クラウドが提供する開発サービスの活用

アジャイル開発では設計からテストまでの短いプロセスを何度も反復する。そのため，ウォーターフォール型開発に比べ開発コードの構築・展開作業が頻発し，これらの作業の効率化が生産性に直結する。クラウド上のシステム開発であれば，開発環境の利用，ソース構成管理，ビルド自動化，インフラサービスへのコード配置自動化，テスト自動化，継続的インテグレーションといった開発の重要要素をクラウド上のサービスで構成することが可能である(**図3**)。

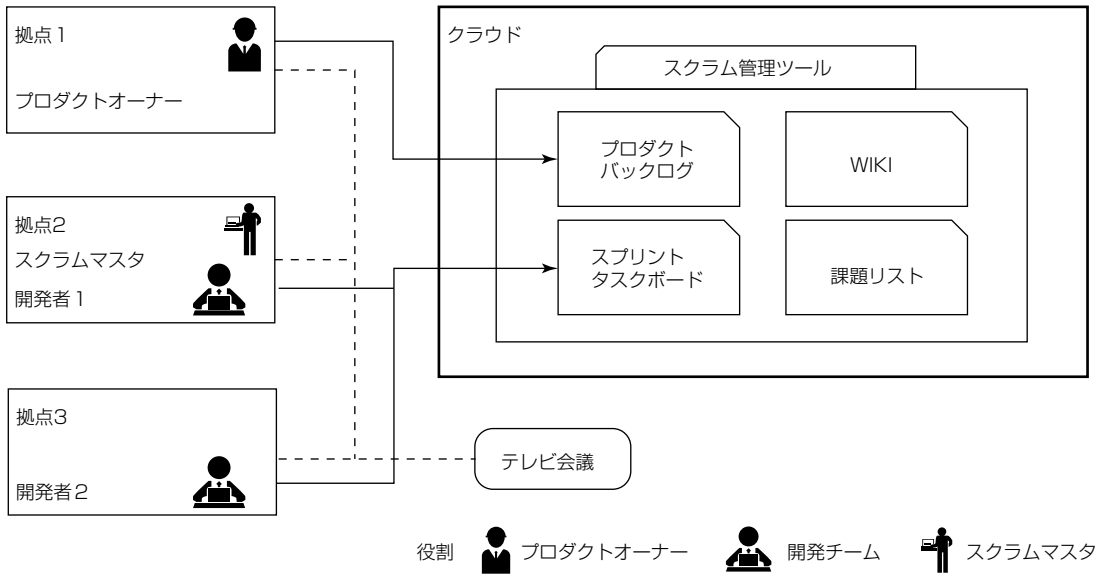


図2．スクラム管理ツールを用いた分散拠点でのスクラム開発

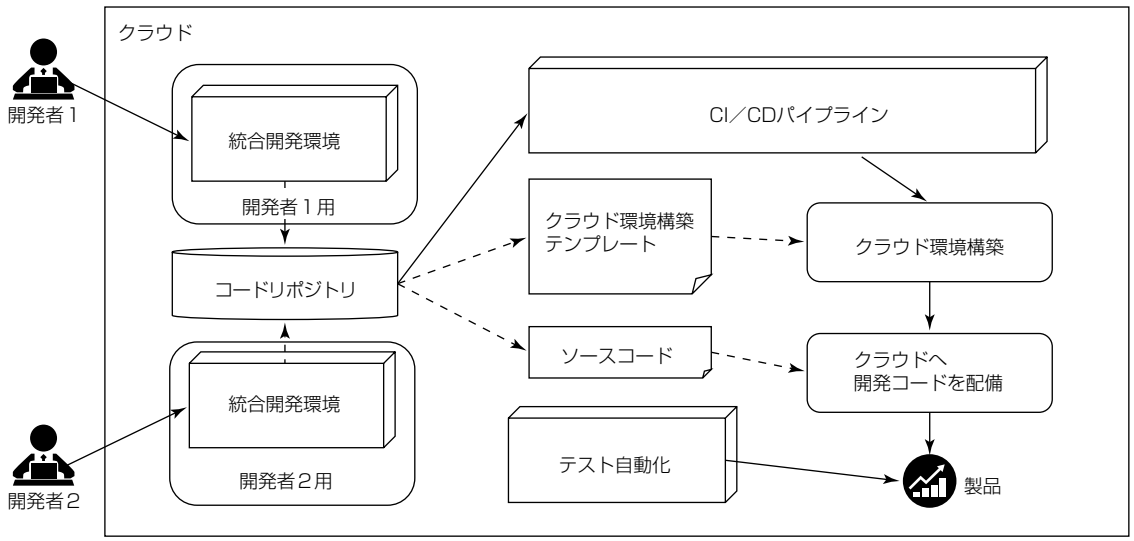


図3．クラウドサービスを活用した開発環境の構成

これは地理的な依存がなく多拠点開発でも有効である。

4.5 アジャイルへの過度な期待を現実化する努力

顧客がアジャイル開発を導入する背景には何らかの“従来型開発手法への不満”があり，その不満を“アジャイル開発を導入すること”が解決すると期待している場合がある。しかし，期待が過度になると，実践後の失望に変わることがある。これはオンプレミス環境のシステム開発からクラウドへのシフトでも生じることであり，“アジャイル開発＋クラウド化”を行うプロジェクトではこの過度な期待が顕著になる。

4.5.1 “アジャイルならすぐに何でもできる”という誤解

アジャイル開発での“動くソフトウェアのリリースを重視する”思想への期待から，新規性が高く開発規模が見えていないサービス企画に対し“アジャイル開発を採用すれば必ず少ないリソース・工期で完成できる”と期待されるケースがある。実際には，開発手法を変えるだけで開発チームの生産性が劇的に向上するわけではない。アジャイル開発を採用するメリットは生産性の向上よりも，早い段階で動作するプロダクトを評価するなど，開発にかかるベロシティから開発完了までの規模を予測し，早期に開発要件を取捨する判断を可能にすることである。

4.5.2 過度な期待を現実的な成功へ導くために

この誤解を解くためには，採用するアジャイル開発の実践方法を説明し，スケジュール内に実現できる要件の総量を予測してベロシティが安定するまでスプリントを繰り返し経験する必要がある。また，達成可能な総ストーリーポイントは“平均ベロシティ×スプリント数”で予測されることを顧客に実感してもらい，“短いサイクルの開発と判断の積み重ね”が迅速に製品を開発する仕組みの実態であることを理解してもらう努力が必要である。

4.6 成果ドキュメントの整備とウォーターフォール型開発との連携での課題

アジャイル開発は製品実装を優先する方針から，ドキュメント整備が疎(おろそ)かになりやすい。回避手段としてプロダクトバックログの完了条件に仕様書作成を盛り込む，製品リリース時にまとめて体系だった文書化をする等の運用ルールを定義する。注意すべき点は，プロジェクトの都合によってプロトタイプ開発をアジャイルで行った後に商用化をウォーターフォール型体制へ移管する場合である(商用化時は社内品質管理ルールに準拠させるためこのケースとなることがある)。本来アジャイル開発では“ドキュメントより動く製品を重視”する方針であり，プロダクトオーナーはスプリントの終盤まで新機能の実装に生産力を投入したくなる。しかし，ウォーターフォール型開発へ引き継ぐ場合は，アジャイル開発完了前に機能実装を止めてでも十分なドキュメント整備時間を確保することで開発手法変更による影響を低減すべきである。

5. むすび

クラウドの台頭によって最新のインフラを少ない投資で開発対象に活用することができるようになった。一方で，サービス提供システム開発は人月ベースの見積りを根拠に，既知の技術を使って全て制御できる前提で計画することが困難になった。この状況下で迅速かつ柔軟なサービス開発を行うための施策としてアジャイル開発が改めて脚光を浴び，MDISは顧客とともにアジャイル開発を用いたクラウド上のサービス開発に取り組んでいる。本稿で述べたとおり多くの利点を享受しつつ従来型開発と異なる課題に直面するなか，サービスインテグレータとして顧客と協調し，アジャイル開発の経験を積み上げ，最新の技術を活用したより良いサービスを提供していく。

参考文献

(1) Beck，K.，ほか：アジャイルソフトウェア開発宣言(2001)
https://agilemanifesto.org/iso/ja/manifesto.html
(2) Rasmusson，J.，ほか：アジャイルサムライ―達人開発者への道―，オーム社(2011)