特集号
招待論文

アジャイルと品質会計
—プロジェクトの高成功率を確保するハイブリッドアジャイルへの取り組み—

誉田 直美 †1

†1 日本電気（株）

日本におけるアジャイル開発の適用は途上の段階にある．NECでは，スクラム，XPおよびウォーターフォールモデル向けの独自品質管理技法である「ソフトウェア品質会計」のプラクティスを組み合わせたハイブリッドアジャイルを考案し，アジャイル開発プロジェクトの成功率を高める工夫をした．このNECアジャイルの内容を紹介するとともに，その適用結果をウォーターフォールモデルの開発結果と比較することにより，NECアジャイルの効果と今後の展望を考察する．

1. はじめに

ソフトウェアの重要性はますます高まり，今やソフトウェアなくして日常生活は成立しない．IoTの進展は，その勢いを加速する．現代社会においては，安心かつ安全で快適な生活への要求とともに，めまぐるしく移り変わる社会へのすばやい対応が強く求められる．これらの要求に応えるコンポーネントがソフトウェアである．すなわち，ソフトウェア開発には，高信頼性の実現と，変化に軽やかに対応する柔軟性の両方が要求される．従来のウォーターフォールモデルでの開発では，計画立案から出荷までの期間が長いため，特に変化に対する素早い対応に課題が見受けられるようになった．代わりに注目されるようになったのが，アジャイル開発である．

2001年にアジャイル宣言が発表されてから，15年が経過した．アジャイル開発の支持者は，プロセス戦争は決着し，アジャイル開発は勝利を得た[1]と言う．一方，日本のソフトウェア開発の現場では，ウォーターフォールモデルが主流のままである[2]．本稿では，アジャイル開発の普及状況や課題を確認するとともに，NECでの取り組みを紹介する．さらに，その適用結果を考察して今後の展望を述べる．

筆者は，NECで長年ソフトウェア開発の品質保証を担当し，近年はNEC全社のソフトウェア開発の品質および生産性向上に取り組んでいる．アジャイル開発としては，本稿で紹介するNECアジャイルの策定をリーダとして推進するとともに，実業務においても，アジャイル開発チームを持ち，プロダクトオーナーとして開発を推進している．

アジャイル開発では，単に短期間に繰り返し開発してリリースするだけでなく，同時に所定の品質を確実に確保することが求められる．本稿では，アジャイル開発を，「所定の品質を確保したソフトウェアを，短期間で繰り返しリリースする手法」と定義する．

以下，本稿は次のように構成する．第2章では，アジャイル開発の現状と課題を述べる．第3章では，NEC独自の品質管理技法であるソフトウェア品質会計（以降，品質会計と呼ぶ）の概要を説明した上で，NECアジャイルを説明する．第4章では，NECアジャイルの適用結果を紹介し考察する．まとめを第5章で述べる．

2. アジャイル開発の現状と課題

2.1 他国と比較した日本の状況

海外でのアジャイル開発の普及状況を図1に示す．これは，2014年の年次調査結果であり，2014年が9回目の調査である．65%が北アメリカ，21%がヨーロッパと主に欧米を対象とした3,925件の回答に基づく．回答者はプロジェクトマネージャ，開発者，開発リーダが60%

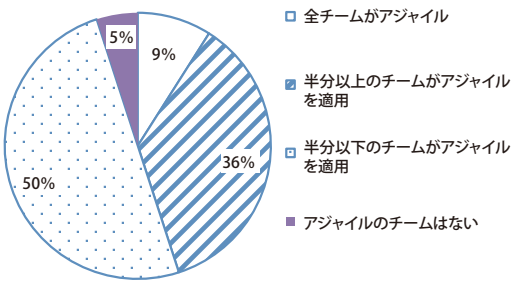

図1　アジャイル開発を利用するチームの割合（文献[3], p.6のデータに基づき筆者が作成）

2016 ©Information Processing Society of Japan

を占める．この調査結果によると，95％が自社にアジャイルチームを保有し，アジャイルのチームがないと回答したのはわずか5％である．これより，欧米では，アジャイル開発が一般的になってきていると考えられる．

次に，アジャイル開発方法論で最も有名なスクラムの有資格者数の各国比較をする（表1）．このデータは2012年調査のものである．米国の資格者数が76,100人と群を抜いて多く，英国13,600人，中国4,230人などである．これに対して日本はわずか476人で，米国の1％にも満たない．この数字から，日本ではアジャイル開発の普及が遅れていることが分かる．

次に，日本と他国のソフトウェア開発プロジェクト構成を比較する．米国では，72％のIT技術者がユーザ企業に所属している（表2）[2]．このため，米国では顧客と開発チームが一体となって開発しやすく，アジャイル開発を適用しやすい環境にあるといえる．一方，日本を含む他国はまったく逆の比率であり，70％強のIT技術者がITサービス企業に所属する．したがって，日本ではアジャイル開発が前提とする顧客と開発チームが協調する開発体制をとりにくい．また，日本ではITサービス企業がユーザ企業から開発を委託される請負開発が一般的である．あらかじめ開発物件を決めて開発を委託する請負開発に慣れている顧客にとって，開発しながら要件を決めていくアジャイル開発は，手間がかかると感じられる．また，アジャイル開発は，請負開発で一般的に適用される一括請負契約とも整合しない．

2.2　アジャイル開発の課題

次にアジャイル開発そのものの課題について考える．アジャイル開発宣言から15年が経過し，欧米ではアジャイル開発の課題が現場でかなり認識されるようになった．アジャイル開発が失敗する原因を表3に示す．これは，上述の図1と同じ欧米を中心とした調査結果に基づくものである．

これによると，1位は，アジャイル手法の経験不足である．同様の原因として，トレーニングが不十分（8位），チームがアジャイル開発に乗り気でない（7位）と2つ入っている．開発者のスキルや意識に関係する課題は，ウォーターフォール開発でも常に問題になるが，アジャイル開発でも同様ということが分かる．ただし，アジャイル開発特有の背景として，アジャイル開発向けに多数のプラクティスが提案されていて修得に時間がかかる上，現場での適用に迷い，効果のある適用方法にたどり着くまで時間がかかることが考えられる．

2位には，アジャイルの価値と企業原理・文化がかみ合わないという原因があがっている．同様の要因としてマネジメント側と思われる原因が3つあることも注目すべきである（3位，4位，5位）．米国のように，アジャイル開発がかなり普及し，理解されている社会で，アジャイル開発が企業原理やマネジメントとかみ合わないことが失敗の原因の上位に位置することに注目すべきである．このような指摘はウォーターフォール開発ではあまり見られない．その理由として，ウォーターフォール開発に代表される計画駆動型プロセスは，あらかじめ計画を立案して実行するため，計画的に業務遂行しやすくマネジメントと親和性があるのに対して，アジャイル開発に代表される実績駆動型プロセスは，実行した結果に基づいて次の策を決定するため，将来を見通した計画を立案しにくく，投資効果をあらかじめ評価することが困難であることが考えられる．変化へ柔軟に対応するためのアジャイル開発の特性が，ある面ではマネジメントの課題となっていると思われる．

組織の大きさに関する指摘（6位）も考慮する必要がある．アジャイル開発チームは，10人以下の小チームで実施されることが多く，スケールアップの問題はソフトウェア開発として大きな課題である．その解決のため

表1　各国のスクラム有資格者数 [2]

	米国	英国	中国	ブラジル	日本
CSM ☆1	67,000	11,800	3,800	4,600	350
CSPM ☆2	8,000	1,800	400	900	120
CSP ☆3	1,100	0	30	60	6
合計	76,100	13,600	4,230	5,560	476

表2　各国の所属企業別比率（文献 [2] のデータに基づき筆者が作成）

	米国	中国	ブラジル	日本
IT サービス企業	28％	72％	75％	75％
ユーザ企業	72％	28％	25％	25％

☆1 CSM（Certidied Scrum Master）：チーム全体の支援者
☆2 CSPM（Certified Scrum Product Owner）：製品の責任者
☆3 CSP（Certified Scrum Professional）：スクラムの実践者

表3　アジャイル開発で失敗する原因 [3]

No.	比率	失敗原因
1	44%	アジャイル手法の経験不足
2	42%	アジャイルの価値と，企業理念・文化が噛み合わない
3	38%	マネジメントの支援不足
4	37%	従来のウォーターフォールプロセスに従うべきという外部圧力
5	36%	文化を変えるための支援不足
6	33%	組織の大きさ，すなわちコミュニケーションの問題
7	33%	チームがアジャイル開発に乗り気でない
8	30%	トレーニングが不十分
9	6%	不明

※重複回答可

に，複数の小チームを構成する場合には，チーム間のコミュニケーションが問題となることが多い．スケールアップ対応のために提案されている技法として，Scrum of Scrums[4]，SAFe[1]，DAD[5]などがあるが，現時点は技法の提案段階との認識である．

日本でのアジャイル開発適用においては，上述した課題に加えて，品質確保への不安を加える必要がある．他国と比較して日本の品質要求が高いことは，よく知られている．事実，日本の出荷後品質バグ数は他国と比較して少ない[6]．アジャイル開発を適用することによって，品質が確保できなくなる事態は避けねばならない．

以上をまとめると，アジャイル開発そのものの課題としては，以下の4点が考えられる．

① 多数のプラクティスが提案されているため，現場で効果的な適用方法を得るまで時間がかかる
② 品質確保に不安がある
③ 投資効果の判断や将来を見通した計画立案をしにくい
④ 開発チームのスケールアップが難しい

本稿では，④を除く①～③の解決に注目して，その解決策を述べる．

3. アジャイルと品質会計

NECでは，NEC独自のソフトウェア品質管理技法である品質会計[7]を，全社的に適用している．品質会計は，20年以上にわたるウォーターフォールモデルでのソフトウェア開発の品質確保への取り組みの結果，構築した．NECでは，アジャイル開発の広がりに対応して，品質会計の品質確保の考え方をアジャイル開発へ適用した社内標準を2011年に策定した．これは，ソフトウェア開発の品質確保の本質は，プロセスモデルによらず同じであるという考えに基づく．この社内標準を「NECアジャイル」[8]と呼ぶ．本章では，はじめに品質会計の特徴を紹介し，続けてNECアジャイルを説明する．

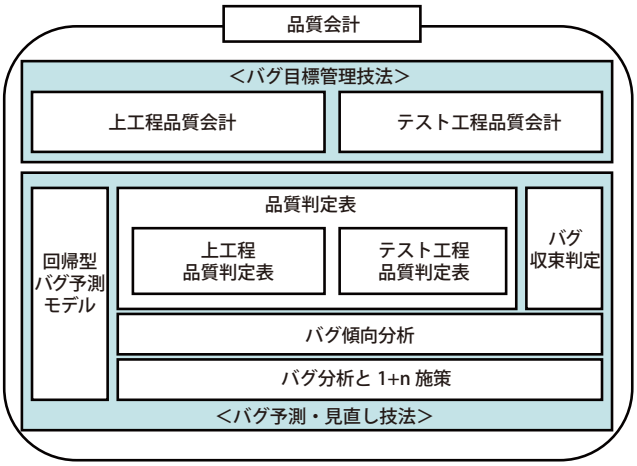

図2　品質会計を構成する技法

3.1　品質会計

品質会計の想定する開発プロセスはウォーターフォールモデルである．品質会計を適用することにより，所定の品質を作り込んだことを，根拠をもって説明できるようになる．品質会計で使用する主要メトリクスはバグ件数であり，バグ件数の摘出目標管理を主軸として品質管理を実施する．開発開始時の摘出バグ件数の目標値設定に始まって，開発途中の各段階での品質分析および摘出バグ目標値見直しで使用する技法を図2に示す．また，開発プロセスと各技法の適用関係を図3に示す．図3の上工程はウォーターフォールモデルの設計および製造工程（以降，設計・製造工程と記す），テスト工程は単体テスト以降のテスト工程を指す．

品質会計の品質確保の考え方の特徴は以下の3点である．

① レビューでのバグ摘出による早期品質確保

設計・製造工程でのレビューにより，出荷までに摘出する全バグの80％摘出を目標とする．これは，テスト段階でのバグ摘出に比べて，設計・製造工程でのバグ摘出のほうが品質および生産性向上に効果的なためである．

② 確実なテスト完了判断

実施すべき項目をすべて完了したことを確認するために，バグ傾向分析，バグ分析と1+ n施策およびバグ収束判定の3つの技法[7]を使用して，テスト完了を判断する．テスト完了とは，実施すべき項目をすべて完了したということであり，バグはもう検出されないことを意味する．

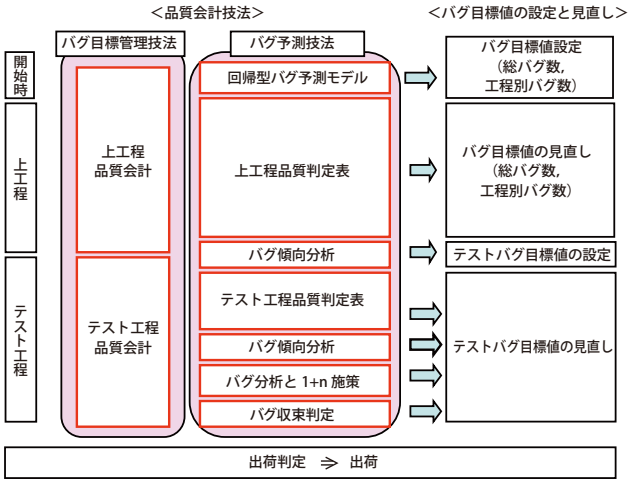

図3　開発プロセスと品質会計技法の適用の関係

③ 品質会計を軸とした全体的な仕組みの構築

品質会計の適用により得られるデータの分析により，自組織の強みと弱みを把握することができる．その強みと弱みを考慮してソフトウェア開発の全体的な仕組みを構築する．この仕組みを運用し改善することにより，ソフトウェア開発プロジェクトの成功率を向上させることができる．

3.2　NEC アジャイルの目的と概要

NECアジャイルの目的は，アジャイル開発適用時の高いプロジェクト成功率の確保である．NECアジャイルは，スクラム[9]，XP[10]および品質会計のプラクティスを組み合わせたハイブリッドアジャイルである．

NECアジャイルの基本的な枠組みは，立ち上げ，基本サイクル，出荷判定の3つのフェーズにより構成される（図4）．立ち上げでは，アジャイル開発体制の構築，開発環境や規約などの検討と設定，アーキテクチャ設計などを実施する．基本サイクルは，3回の開発スプリントと1回のリリーススプリントを1回のセットとして実施する．基本サイクルを1回終了すると，出荷可能なソフトウェアができあがる．このソフトウェアを出荷する場合は，出荷判定フェーズで出荷判定を実施し，合格後，出荷する．出荷の必要がなければ，出荷判定をせずに基本サイクルを繰り返す．

1回のスプリント期間は原則として2週間，1回の基本サイクルは8週間である．したがって，NECアジャイルでは，8週間ごとに出荷可能なソフトウェアを開発することができる．

3.3　各フェーズでの実施項目

各スプリントでの実施項目を表4に示す．また，品質会計とNECアジャイルとの関係を図5に示す．

立ち上げでは，アジャイル開発チームを結成し，要件の整理，大まかなスケジュールの検討，開発ルールの整備およびアーキテクチャ設計などを行う．本格的な基本サイクルでの開発へ入る前に，全体のアーキテクチャを設計しておくことにより，保守性の低いソフトウェアにならないようにする．

次の開発スプリントでは，優先順位の高い開発項目（以降，ストーリーと呼ぶ）を選んで，設計，製造およびテスト（システムテストを除く）を実施する．ストーリーは，数日で開発できる小さい開発項目である．リリーススプリントは，3回分の開発スプリントで開発したストーリーに対して，システムテストおよびマニュアル作成などの，リリースに必要な作業を実施する．開発スプリント3回＋リリーススプリント1回を1回の基本サイクルとしている理由は，デグレードの早期摘出とシステムとしての品質確保である．アジャイル開発は小さい開発項目であるストーリーを順次開発していくため，デグレードのリスクが高い．その摘出が遅れると品質確保に時間がかかり，アジャイル開発の最大の目的である動くソフトウェアの提供へ大きな影響を及ぼしてしまう．

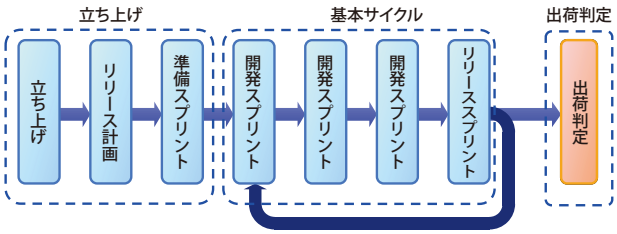

図4　NEC アジャイル

表4　各フェーズの実施項目

フェーズ		目的	実施項目	必須プラクティス
立ち上げ	立ち上げ	プロジェクト立ち上げ	• 開発体制の構築 • 役割の決定	なし
	リリース計画	リリース計画立案	• 要件の整理 • 開発ルールの整備 • 大まかなスケジュールの検討	なし
	準備スプリント	開発開始のための準備	• アーキテクチャ設計 • 開発環境の準備 • 各種規約準備	• スプリントレビュー
基本サイクル	開発スプリント	ストーリーごとの開発	• 要件の詳細定義 • 設計および実装 • 単体テスト • 機能テスト ※必要な仕様書を作成	• ペアワーク • 継続的インテグレーション＋自動テスト • ソースコードの共同所有 • 集合レビュー • スプリントレビュー
	リリーススプリント	システムテストおよびリリース作業	• リグレッションテスト • システムテスト • マニュアルの作成 • リリース物の配布準備 ※必要な仕様書を作成	品質会計の以下の技法 • バグ傾向分析 • バグ分析と1+n施策 • バグ収束判定 • スプリントレビュー
出荷判定	出荷判定	出荷判定	• 出荷判定	• 出荷判定基準に基づく出荷判定

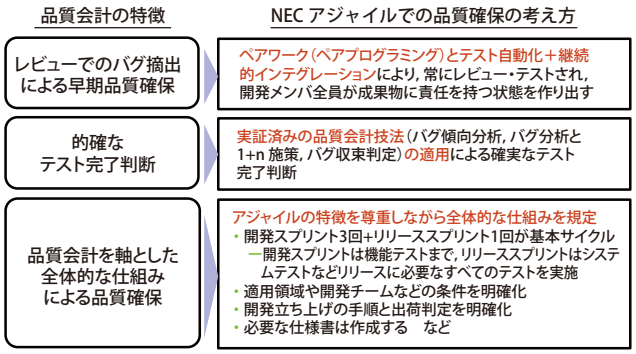

図5　品質会計と NEC アジャイル

開発スプリントでは，品質会計の「レビューでのバグ摘出による早期品質確保」という考え方に基づき，常にレビューおよびテストされた状態を作るために，ペアワークおよびテスト自動化+継続的インテグレーションを必須プラクティスとしている．必須プラクティスとは，NECアジャイルで適用必須と定義したプラクティスである．品質会計の品質確保の考え方と，各プラクティスの実施効果を考慮し，効果があると判断できるプラクティスとその適用のヒントをあらかじめ提示することにより，現場での無用な試行錯誤を避ける工夫をしている．ペアワークとは別に集合レビューも実施する．これは，開発チーム全員で合意し共有すべき情報を，集合レビューで確認するためである．ペアワークで共有できる情報はペアに限定されるため，集合レビューは欠かせない．

リリーススプリントでは，品質会計の「的確なテスト完了判断」に基づき，品質会計の3つの技法（バグ傾向分析，バグ分析と1+ n 施策およびバグ収束判定）をそのまま適用している．これにより，バグが残存していないことを確認する．

開発スプリントおよびリリーススプリントの終了時には，スプリントレビューを実施し，スプリント完了基準（図6）に基づいてストーリーの終了判定をするとともに，振り返り（KPT[11]）を実施する．

出荷判定では，ウォーターフォール開発での出荷判定項目と同等の項目により，出荷の合否を判断する．

NECアジャイルでは，上述のようなアジャイル開発の全体的な枠組みを構築することにより，アジャイル開発の最大の利点であるアジリティを確保しながら，品質確保に必要な作業が確実に実施できるように工夫した．これは，品質会計の「品質会計を軸とした全体的な仕組みの構築」の考え方に基づくものである．

3.4　NECアジャイルでの役割

NECアジャイルで定義する役割は4種類ある（表5）．

- ✓ すべてのソースコードが以下の閾値を満足している
  - ✓ 関数の有効行数　●●行未満
  - ✓ 複雑度　○○以下
  - ✓ ネスト　○○以下・・・
- ✓ C0コードカバレッジ　100%
- ✓ 単体テストコードが自動化された環境上で実行完了　100%
- ✓ 機能テストコードの実行完了　100%
- ✓ すべてのソースコードを作成し，構成管理上にチェックインしている
- ✓ すべてのテストコードを作成し，構成管理上にチェックインしている
- ✓ すべての設計仕様書を作成し，構成管理上にチェックインしている
- ✓ すべてのテスト仕様書を作成し，構成管理上にチェックインしている

・
・
・

図6　開発スプリントの完了基準（例）

スクラムで定義するプロダクトオーナー，開発チーム，スクラムマスターの3つの役割に加えて，品質技術者を定義している．品質技術者は，開発チームの開発状況を監視し，品質の作り込み状況を客観的に確認することにより品質を保証する．具体的には，日常的に開発業務を監視することにより，開発進捗に応じて作成される設計結果やプログラムなどの成果物の内容や，工数やテスト項目数などのデータをチェックする．不十分と思われる場合は，その場で開発チームへ是正を依頼する．各スプリント終了時には，開発途中の品質確保状況に基づいて，プロダクトオーナーのスプリント完了判断を補佐する．したがって，品質技術者には，単に数値確認だけではなく，開発の中身の妥当性を確認できる能力が求められる．

3.5　スプリント完了基準

スプリント完了基準とは，当該スプリントで実施すべき項目の完了を判断するための基準である．完了スプリントおよびリリーススプリントに対して各々完了基準がある．毎回のスプリント終了時に，スプリント完了基準に基づいて完了を判断する．スプリント完了基準を満足しなければ，実施すべき項目を完了していないものとして，開発を継続する．

スプリント完了基準は，できるだけ具体化して定量的に表現することが肝要である（図6）．その理由は，スプリント完了基準が，開発の具体的な指針となるためである．スプリント完了基準が具体的であるほど，開発作業への要求事項が具体化され，開発作業内容に関する理解の齟齬がなくなる．たとえば，図6の例によると，開発するソースコードは，関数の有効行数，複雑度，ネストの各閾値を満足し，単体テストは100％自動化して，C0コードカバレッジ100％を満足しなければ完了と判断しない．このように，スプリント完了基準は，できるだけ判定者に依存せずに，同じ判断結果が得られるような完了基準が望ましい．

表5　NECアジャイルで定義する役割

役割	担務
プロダクトオーナー	開発するプロダクトの責任者．要件の内容と開発する優先順位を決定する
開発チーム	要件に従って設計・製造・テストを行う
スクラムマスター	プロダクトオーナーと開発チームが正しくスクラムを実践できるよう支援する
品質技術者	品質の作り込み状況を客観的に確認する • Doneの定義の満足状況を，実物を見ながら確認 • データにより開発状況を確認 • 顧客視点での第3者評価

3.6　収集データ項目

NECアジャイルで収集するデータ項目を表6に示す．

アジャイル開発は，ウォーターフォールモデル適用時に比べて，かなり詳細な単位でデータを収集できる．その理由は，アジャイル開発では通常，1～2時間単位まで詳細化したタスクの単位で作業を実施するため，そのタスク単位でデータが収集可能になるからである．

NECアジャイルでは，バグの計測対象は，当該ストーリーの開発スプリントの完了判定以降に摘出されたバグと定義している．その理由は，設計，製造およびテストを同時に進めていくアジャイル開発では，開発スプリントの完了判定前は設計の試行錯誤段階とみなすのが妥当と考えているためである．ウォーターフォールモデルの品質会計でも，設計途中の試行錯誤をバグの計測対象としていない．したがって，NECアジャイルでの計測対象バグは，ほかのストーリーを作り込む際のデグレードか，システムテストでなければ摘出できないようなバグだけであり，ウォーターフォール開発でのバグ計測対象に比べてきわめて限定的である．

NECアジャイルでは，カバレッジや複雑度といった，ツールを適用しなければ計測が難しいメトリクスを収集している点も特徴的である．アジャイル開発は，その開発プロセスの性格上，ツールなどの開発環境が整備されている必要がある．

表6　収集データ項目

No.	種別	メトリクス	単位
1	工数	設計・実装	人H
2		テスト	人H
3		リファクタリング	人H
4		マニュアル作成	人H
5		ペアワーク	人H
6		集合レビュー	人H
7	ストーリー	完了	件
8		申し送り	件
9	ベロシティ☆4	全体ストーリーポイント	SP☆5
10		消化ストーリーポイント	SP
11	規模	ソフトウェア	Line
12		テストコード	Line
13	テスト項目数（テスト種別ごと）	自動　（新規・リグレッション別）	項目
14		手動　（新規・リグレッション別）	項目
15	バグ数	全体バグ数	件
16		うちデグレード	件
17		うち重大	件
18	仕様書	仕様書数	ファイル数
19		行数	行
20		図の数	個
21		メモ数	個
22	コードメトリクス	カバレッジ	%
23		ネスト	ネスト数
24		複雑度	複雑度数

☆4 ベロシティ：スプリント当たりの作業量
☆5 SP（ストーリーポイント）：相対見積もり値

4.　適用結果と考察

NECアジャイルは，考案から4年が経過し，ようやく安定的に適用され，プロジェクトを横串にしたデータ収集が可能となった．現在は，プラクティスの適用方法などの細かいレベルで見直しをしている段階である．

本章で紹介するデータは，NECグループ内の2014年度に出荷したソフトウェア開発プロジェクトのデータである．現時点ではアジャイル開発がWebシステム開発を中心に適用されているため，アジャイル開発のデータは，Webシステム開発プロジェクト6件である．ウォーターフォール開発のデータは，Webシステム開発以外も含む，一般的なシステムインテグレーション開発358件で，アジャイル開発の平均的な開発規模と同程度の開発規模のプロジェクトに限定している．使用する数値は，データ各層の中央値である．

4.1　生産性

図7に単位規模あたりの開発工数を示す．アジャイル開発の工数は，ウォーターフォール開発の約1/2程度である．また，工数を設計・製造工数とテスト工数に分けると，設計・製造工数とテスト工数のいずれも，ウォーターフォール開発と比較してアジャイル開発のほうが少なく，特に設計・製造工数は，かなり少ないことが分かる．したがって，生産性は，ウォーターフォール開発よりもアジャイル開発のほうが良い．これより，NECアジャイルでは一定レベルの生産性を確保できているといえる．

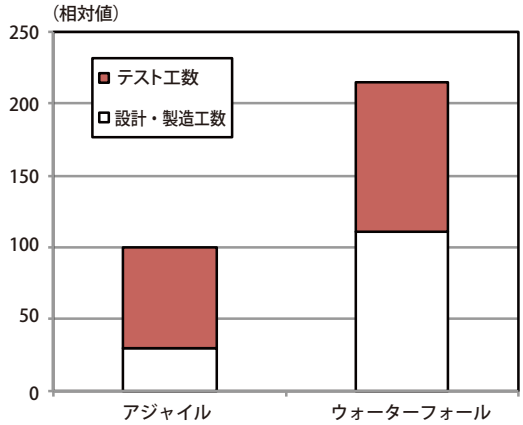

図7　単位規模あたりの開発工数

アジャイル開発の設計・製造工数が少ない理由は，開発単位が小さく，数日で一気に開発する繰り返しであるためと考える．開発する単位は，小さいほうが範囲が限定されるため，設計しやすく細かな個所を考慮しやすい．しかも1つのストーリーを，設計からテストまで数日で一気に実施するため，開発効率が良い．また，開発スプリント中に開発者が入れ替わることはほとんどない．一方，ウォーターフォール開発では，全開発項目を対象に，設計，製造，テストと開発を進めていく．工程により開発者が入れ替わることも多い．このような開発条件の違いが，生産性に影響しているものと思われる．

設計･製造とテストにかかる工数の比率を比較すると，アジャイル開発では設計・製造工数：テスト工数≒3：7，一方，ウォーターフォール開発では，設計・製造工数：テスト工数≒5：5である（図7）．アジャイル開発で設計・製造よりもテストに工数がかかる理由は，テスト自動化を前提として，自動化したテストを繰り返し実行することにより，ソフトウェアが動くことを保証する開発方法だからである．テスト自動化は，テスト実施には工数がかからないものの，テスト自動化そのものを維持するには，多くの労力がかかる．これは，新しいストーリーを開発するたびに，自動実行できるよう，関係する全テストプログラムを修正しなければならないためである．また，頻繁にソフトウェアを出荷するため，出荷に必要なシステムテストを出荷のたびに実施する必要がある．これは，システム規模の大きさによっては，かなりの負担になるはずである．一方，ウォーターフォール開発では，効果のある範囲を見極めてテスト自動化を実施する．多くのテストは新たにソースプログラムを変更しない限り，当該開発では一度実行されるだけである．

4.2　品質

出荷後バグ数というメトリクスでの計測結果では，今までのところ，アジャイル開発で出荷後バグは発生していない．その意味で，NECアジャイルの品質確保という目的は達成している．ただし，現時点においてアジャイル開発が，Webシステム開発が中心で，使用性を重視したプロジェクトが多いためという適用条件による影響も大きいと考える．アジャイル開発は，要件との食い違いを，動くソフトウェアで確認して，次のスプリントで修正するか，納得してそのまま出荷するため，もともと出荷後バグとして指摘されにくい手法と思われる．

次に，アジャイル開発の出荷前までの品質作り込みについて，ウォーターフォール開発と比較しながら考察する．ウォーターフォール開発の品質会計は，何重かのさまざまな立場の技術者による技術レビューや工程移行時のマネジメントレビューなどの仕組みを構築し，問題が検出されれば同種の問題が内在していないかを確認する，重層構造の仕組みで所定の品質を確保する．

これに対して，NECアジャイルの仕組みは，スプリントレビューでのスプリント完了基準に基づく完了評価，およびスプリント中の品質技術者による品質確認であり，ウォーターフォール開発の仕組みと比較してきわめて軽い．NECアジャイルの品質確認の方法は，設計内容やテスト内容の具体的な中身の妥当性の判断が中心である．ウォーターフォール開発の品質会計で，数値に基づく品質分析をした後，設計の中身の分析に入るのと異なる．これは，NECアジャイルで計測対象とするバグ数の件数がきわめて少ないため，数値よりも内容の分析が主になるからである．

ソフトウェア開発で開発者のスキルが成果物の出来に大きく影響を及ぼすことはよく知られている[12]．ウォーターフォール開発の重層構造の仕組みは，開発者のスキルの影響を受けにくくするための工夫である．NECアジャイルでも，同様の考え方で仕組みを構築しているものの，アジリティを尊重するために，ウォーターフォール開発のような重層構造にはしていない．仕組みを重層化する代わりに，一定スキルを身に付けた開発者がアジャイル開発に携わるように考慮している．

3.3節で述べたように，アジャイル開発の品質リスクとして検討すべきは，デグレードである．デグレードは，改造による影響範囲の見誤りに起因することがほとんどであるため，影響範囲を想定した範囲でテストを実施する開発スプリントでは検出が難しく，リリーススプリント時に，広範囲を対象にしたシステムテストで摘出されることが多い．NECアジャイルで開発スプリント3回+リリーススプリント1回を基本サイクルとする方法は効果を上げており，リリーススプリントでのデグレード摘出は複数回に及んでいる．

4.3　NECアジャイルの成果と今後の展望

NECアジャイルは，基本的な枠組みを考案し，各スプリントで実施すべき項目，収集メトリクス，登場人物の役割等を定義するとともに，品質会計で培った品質確保のポイントを組み込んで構築した．その結果，一定レベルの品質や生産性を達成していることを，4.1節および4.2節で示した．これらにより，第2章でアジャイル開発の課題としてあげた，①多数のプラクティスが提案

されているため，現場で効果的な適用方法を得るまで時間がかかる，および②品質確保に不安があるという2つの課題を解決した．また，NECアジャイルにより基本的な枠組みを設定することで，複数のアジャイル開発プロジェクトの横串でのデータ分析が可能となり，アジャイル開発の平均的な生産性や品質のベースライン値を提供できるようになった．これにより，③投資効果の判断や将来を見通した計画立案をしにくいという課題についても，一定の解を示すことができている．

残されたNECアジャイルの課題は，スケールアップへの対応である．これは，ウォーターフォール開発とアジャイル開発を並行して進めるなど，ウォーターフォール開発の知見の取り込みにより解決する見通しである．

5. おわりに

本稿では，アジャイル開発の現状と課題を述べ，NECでのアジャイル開発への取り組みと成果を紹介した．

アジャイル開発かウォーターフォール開発かという優位性を論ずることは意味がなく，各々の特性を理解した上で適材適所で適用することが求められる．アジャイル開発は，メリットだけが先行する段階を過ぎて，課題もかなり明らかになってきている．その課題解決に対しては，個々の開発チームで取り組むだけでなく，組織として取り組むべきである．それが，アジャイル開発での成功率を向上するために必要だからである．アジャイル開発の仕組みは，アジリティを活かすため，必然的に軽くなると思われる．軽い仕組みを補完するために，個々の開発者には一定レベルのスキルが求められるようになる．これは，ソフトウェア業界へ良い効果をもたらすものと考える．アジャイル開発の広がりによって，IT業界がより魅力ある産業になることを期待している．

参考文献

1) Leffingwell, D.（著），藤井　拓（監訳）：アジャイルソフトウェア要求　チーム，プログラム，企業のためのリーンな要求プラクティス，（株）翔泳社（2014）．
2) （独）情報処理推進機構　技術本部 ソフトウェア・エンジニアリング・センター：非ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査報告書（非ウォーターフォール型開発の海外における普及要因編），http://www.ipa.go.jp/sec/softwareengineering/reports/20120611.html#L1　（2016年1月31日現在）
3) VERSIONONE: State of Agile Survey 9th annual, http://info.versionoNECom/state-of-agile-development-survey-ninth.html　（2016年1月31日現在）
4) Schwaber, K. and Beedle, M.（著），スクラム・エバンジェリスト・グループ（訳）：アジャイルソフトウェア開発スクラム，ピアソン・エデュケーション（2003）．
5) Ambler, S. W. and Lines, M.（著），藤井智弘（監訳）：ディシプリンド・アジャイル・デリバリー　エンタープライズ・アジャイル実践ガイド，（株）翔泳社（2013）．
6) Cusumano, M.: Software Development Worldwide - the State of the Practice, IEEE Software (2003).
7) 誉田直美：ソフトウェア品質会計－NECの高品質ソフトウェア開発を支える品質保証技術－，（株）日科技連出版社（2010）．
8) Kosaki, M., Kosumi, Y., Terada, T. and Honda N.: Toward High-quality Agile Software Development: NEC's Agile Development Management Method, ProMAC2013 (2013).
9) Rasmusson, J.（著），西村直人（監訳）：“アジャイルサムライ－達人開発者への道－”，（株）オーム社（2011）．
10) Kent Beck, 長瀬嘉秀（監訳）：XPエクストリーム・プログラミング入門－変化を受け入れる　第2版，ピアソン・エデュケーション（2005）．
11) Cockburn, A.（著），長瀬嘉秀，今野　睦（監訳）：アジャイルソフトウェア開発，ピアソン・エデュケーション（2002）．
12) Boehm, B. W.: Software Engineering Economics, Prentice-Hall (1981).

誉田 直美（正会員）n-honda@ay.jp.nec.com
日本電気（株）入社後，IT系汎用ミドルソフトウェア製品の品質保証に従事．現在はNEC全社のソフトウェア品質向上に従事．主席品質保証主幹．工学博士．

採録決定：2016年4月22日
編集担当：位野木万里（工学院大学）