ソリューション技術通信　ベリサーブナビゲーションでは、さまざまな分野で検証に携わるベリサーブが「今とこれから」を皆様にお届けします。

VERISERVE NAVIGATION

2018
November
Vol.15

巻頭特集

IoTを狙う脅威への対応

IoTセキュリティ評価のための チェックリストをつかった取組み

輿石 隆 氏

巻頭特集

IoTを狙う脅威への対応

IoTセキュリティ評価のためのチェックリストをつかった取組み

興石 隆氏
こしいし たかし

一般社団法人 JPCERT
コーディネーションセンター
早期警戒グループ
情報分析ライン
情報セキュリティアナリスト

はじめに

IoTを始めとした技術の進化に伴い、セキュリティインシデントは日々巧妙化し、実社会に深刻な影響を与えています。日本における情報セキュリティ対策の向上を目的とする組織であるJPCERTコーディネーションセンター（JPCERT/CC）は、2018年11月（予定）にIoTセキュリティ対策の新たな評価ツールとして「IoTセキュリティチェックリスト」をリリースいたします。

本稿では、IoTシステムにかかわる脅威や問題についてご紹介し、その脅威・問題を解決するために参考となる資料としてJPCERT/CCで作成した、IoTセキュリティチェックリストについてご紹介いたします。

1. IoTにまつわるインシデント

近年、IoT（Internet of Things）と呼ばれる仕組みが世界的に注目を集めています。センサーから収集した情報や、デバイスの操作がネットワークを介して容易に実行できたり、複数のシステムやデータが連携しやすくなったりと、インターネットの活用の幅が広がります。

一方で、従来ネットワークに接続されることを想定されていなかったセキュリティの設定の甘い機器がネットワークに接続されたことにより、IoT機器やIoTシステムがサイバー攻撃に巻き込まれる事案が出始めています。国内外で被害が確認されており、場合によっては、攻撃者に利用されて、他のシステムへの攻撃に加担させられてしまうこともあります。本章では、対策の甘いIoT機器が悪用されたサイバー攻撃の代表的な例として、IoTボットネットを形成するマルウエア“Mirai”についてご紹介します。

1-1 IoTボットネットを形成するマルウエア“Mirai”

2016年9月20日、コンピュータセキュリティ情報サイトの1つである「Krebs on Security」やフランスのWebホスティング会社であるOVHをターゲットに前例のない規模のDDoS攻撃[*1]が行われ、サービスの運営に支障をきたしました。また、同年10月22日には、米国でDNSインフラサービスを提供しているDynamic Network Services Inc. (別称Dyn)に対しても、同規模のDDoS攻撃が行われ、Dynを利用している多くのWebサイトやオンラインサービスが影響を受けました。3社の受けたDDoS攻撃におけるデータ転送量は、最大1Tbpsに達するほどで、“Mirai”と呼ばれるマルウエアに感染した機器から構成されたボットネットからデータが送り付けられたものと分析されています。

ボットとは、ネットワーク経由で遠隔からコマンドを受け取り実行する悪意のあるプログラムであり、“Mirai”はその一種です。ボットに感染した機器は、攻撃者の手先として遠隔からの指示に従って機能します。ボットネットとは、ボットに感染した機器を束ねてネットワーク化したものです。攻撃者はコマンド&コントロールサーバ(C&Cサーバ、C2サーバ)を通じて、ボットネットを構成する多数の機器に一斉にコマンドを送ることができ、強力なDDoS攻撃などを起こすことができます。

“Mirai”は2016年8月、マルウエア調査報告グループMalwareMustDie によって、初めて発見、報告されました。その後、2016年10月1日に、世界中のハッカーが意見交換を行うWebサイトの一つであるハッキングフォーラム上にて、“Mirai”のソースコードが公開されていることが報じられ、世間の注目を浴びました。

1-2 “Mirai”の感染

“Mirai”はLinuxおよびUNIXベースのシステム用の共通ファイル形式であるELF(Executable and Linkable Format)形式にて作成されています。このファイル形式は、ルータ、DVR、IPカメラを含む多くのIoTデバイスのファームウェアで使用されています。

“Mirai”は感染を拡大するために、SSHまたはTelnetネットワークプロトコル[*2]を用い、デフォルトおよびハードコードされた認証情報を利用、またはブルートフォース攻撃[*3]を使用してLinuxベースのデバイスにログインします。

そのため、工場出荷時に機種共通の認証情報が設定された状態のままインターネットに接続されて使われていた世界中のネットワークカメラなどの機器が“Mirai”に感染し、膨大な数のボットで構成されるボットネットを形成して攻撃に加担していました。米国のDynの攻撃では、数千万台の機器からの通信がほぼ同時に発生していたと報告されています。

1-3 IoT機器を狙った攻撃の変化

2016年に“Mirai”が注目されたことにより、IoTのセキュリティに関する関心が世の中で高まり、デフォルトの認証情報のまま運用される機器が徐々に減っていきました。しかしながら、その後もIoT機器を狙うマルウエアは、感染させる方法を変えて活動を継続していることが確認されています。

それは、機器固有の脆弱性を利用して感染させる方法です。

“Mirai”のソースコードを流用して作成された、いわゆる“Mirai亜種”の一つで2017年に確認された“Satori”では、ルータ製品などに使われるSoC(System on a Chip)のOS command injectionの脆弱性(CVE[*4]-2014-8361)を悪用したコードが組み込まれています。また、2017年9月には、“Satori”と同様に脆弱性を利用して感染を行うマルウエア“IoT reaper”の活動を確認しております。この“IoT reaper”では、さまざまなネットワーク製品の脆弱性を悪用するコードが実装されています。感染させることができる機器を探索して、悪用可能な脆弱性の有無を調べる“IoT reaper”によるものとみられるスキャン活動をJPCERT/CCでも確認しております。さらに、脆弱性を利用して感染を拡大するボットの存在は、2018年にも確認されており、9月には、Apache Strutsの脆弱性(CVE-2017-5638)やセキュリティアプライアンス製品のSonicWall Global Management System(GMS)の脆弱性(CVE-2018-9866)を悪用したコードを実装した“Mirai”の亜種を確認したと報告されており、IoT機器に感染するボットが、新たな機器を感染させる方法を絶えず変化させています。

*1: Distributed Denial of Service attack:複数のマシンより一つのサービスに向けて大量のデータを送信し、トラフィックの増大によるネットワークの遅延、サーバやサイトへのアクセス不能を引き起こす攻撃

*2: ネットワーク上での通信に関する規約を定めたもの

*3: 暗号やパスワードを解読するための手法のひとつ。総当たり攻撃ともいう。

*4: CVE(Common Vulnerabilities and Exposures:共通脆弱性識別子)の略称

2. IoTセキュリティにかかわる脅威・問題点

前章で紹介した“Mirai”等を含め、IoT機器を狙ったマルウエアが継続して登場しています。そういったマルウエアによる攻撃は、IoT機器の設定の甘さを狙ったものが多く、被害に遭わないために利用者、IoT機器やサービスの提供者の双方で次のような対応を心掛ける必要があります。

機器の適切な設定

“Mirai”のようなIoT機器が感染するマルウエアの多くは、デフォルトの認証情報など、その機器のセキュリティに対する設定が甘いところを狙います。

●利用側での対応

マルウエアへの感染を防ぐためには、認証情報をはじめ、機器が公開しているポートやサービスの適切な制限、その他利用できるセキュリティ機能がないかを確認した上で、IoT機器を設定する必要があります。

●IoT機器やIoTサービスの提供者側での対応

設定を行った上で利用を開始するように利用者を誘導することが重要です。そのためには、必ずしも技術の詳細を理解していない利用者が読んでも分かるように、背景や具体的な操作手順などを丁寧に説明するなどの配慮が必要です。可能であれば、購入後の最初の利用に先立って、そのような設定を強制的にさせるような作りにすることも有効と考えます。

脆弱性の対応パッチの適応

認証情報等の設定の甘さを狙う以外の脆弱性を利用した攻撃においても、その攻撃に利用されている脆弱性は、既知のものであり、修正パッチが公開されている場合が多くあります。

●利用者側での対応

修正パッチを適用し、常に最新のバージョンで機器を管理し、脆弱性の悪用を防ぐことも必要なこととなります。

●IoT機器やIoTサービスの提供者側での対応

利用者に対して最新のアップデートの情報の提供を行ったり、場合によってはネットワークに接続されているIoT機器に対しては、自動でアップデートの更新を行う機構の実装をしたりすることも有効と考えます。

しかし、上記のようなセキュリティ対応を行ってとしても、前章で述べたようにマルウエアは次々と手口を変えて攻撃を行うため、個別の事象に都度対応する方法では、担当者の負担が大きい、対応に見落としが発生するなどの問題が発生してしまいます。

3. IoTセキュリティチェックリストの活用

これまで述べてきたようなIoTを巡るセキュリティ上の問題を解消ないし軽減するためには、IoT機器やサービスの提供者と利用者の双方で留意すべき問題が山積しています。JPCERT/CCでは、そうした問題を可能な限り網羅的に分析し、解決のための指針を与えるチェックリストとして取りまとめました。

このチェックリストは、IoTに関連する多様な立場の関係者にご利用いただけるように構成されていますが、以下では、IoT機器やサービスの提供者にご利用いただくシーンに焦点を絞って、利用方法を中心とした概要を紹介させていただきます。

【概要】

IoTセキュリティチェックリストでは、IoTセキュリティへの具体的な対策を検討する上で必要となる、開発時、利用時における作業時の実用性を考慮した具体的な39項目のポイントにまとめたものであり、利用者は実際に手を動かして、各項目をチェックし、自身のIoTシステムのセキュリティ対策の状況を確認することができます。

IoTセキュリティチェックリストのポイントは次の通りです。

- IT、IoTにおいて、共通で求められるセキュリティに関する項目を羅列している
- IoTシステムの機能ごとに確認するポイントを分けている
- 開発者・利用者ごとの確認内容を設定している

3-1 IoTセキュリティチェックリストの見かた

IoTセキュリティチェックリストでは、評価を行いたいIoTシステムのセキュリティ機能ごとにそれぞれ次のような項目が用意されています。(図1-1)(図1-2)

3-2 チェックリスト利用の流れ

IoTセキュリティチェックリストでは、次のようにIoTシステムの評価を行うことを想定しています。

①評価を行うユーザの確認(開発者/利用者)
②評価するIoTシステムの構成を機能ごとに分類
③確認したい項目の選定
④項目ごとに対象となる機能の評価
(ア)チェック欄に評価対象に該当機能があるかのチェック

(イ)チェックがつかない項目について検討し、その結果を理由欄に記載

この中で特に「②評価するIoTシステムの構成を機能ごとに分類」がポイントとなります。IoTセキュリティチェックリストでは、この機能ごとの分類でセキュリティ項目を評価することにより、IoTを構成するシステム全体でセキュリティが担保できているかの確認や、IoTのデバイスなどの個々の部分のセキュリティの機能が十分かの確認ができます。

①	② 大項目	③ 小項目	④ 本項目の目的	⑤ 開発する際に気を付けること	⑥ 利用する際に気を付けること	⑦ チェック欄	⑧ 理由	⑨ 関連するコンポーネント S:Sensor、A:Aggregator、C:Communication Channel、E:e-utility、D:Decision trigger				
								S	A	C	E	D
1	ユーザ管理	アカウントロックアウトメカニズム	第三者が端末を不正に操作できないようにする	連続した規定回数以上のログイン失敗や多重ログインなどの痕跡を確認したら、アカウントをロックし、ログインが不可になる機能を持たせる	アカウントロックに関する設定可能な内容を確認し、自身で設定したとおりにアカウントがロックされるか確認する			○	○		○	○
2		有効期限切れパスワードへの強制失効オプション	一定期間利用されていないアカウントからのログインをできないようにする	設定した有効期限を超過したパスワードを失効させ、アカウントをロックする機能を持たせる	有効期限後にパスワードが失効することを確認する				○		○	○
3		パスワード強度の担保機能	ブルートフォース、辞書攻撃などにより不正にログインされないようにする	複数の文字種の利用や一定文字数以上などの条件を満たしたパスワードのみ登録できる機能を持たせる	条件に満たないパスワードの登録ができないことを確認する				○		○	○
4		パスワードセキュリティオプション(2要素認証など)	第三者がシステムにログインすることを困難にする	パスワードセキュリティオプションを利用できるようにする(例:2要素認証など)	パスワードセキュリティオプションが利用できることを確認する				○		○	○
5		サービスやプロセスを起動するアカウントの権限管理	アカウント毎にサービスやプロセスを動かす権限を限定してインシデント発生時の影響範囲をサービスやプロセスの範囲内におさえる	サービスやプロセスの起動にスーパーユーザを求めない作りにする	サービスやプロセス専用のユーザで動作することを確認する				○		○	○
6		共有ユーザアカウント	用途に応じて適切な権限を付与できるようにする	アカウント管理機能を持たせる	共有するアカウントが適切な権限と共有範囲で利用されていることを確認する				○		○	○
7		管理ユーザへの適切な権限付与	管理ユーザが必要な権限を使えるようにする	権限を管理する機能を持たせる	管理者ユーザが適切な権限を付与されているか確認する				○		○	○
8		一般ユーザへの権限付与機能	ユーザに必要な権限のみを使えるようにする	ユーザに権限を付与できる機能を持たせる	ユーザに権限が付与でき、権限に応じた利用ができることを確認する				○		○	○
9		認可制御機能	役割に応じたアクセス権を付与するできるようにする	アカウントの役割に応じたアクセス権を付与する機能を持たせる	認可されていない情報や機能が利用できないことを確認する				○		○	○
10		サービス連携	ログイン情報が必要以上に他のサービスに渡らないようにする	サービス連携時に他のサービスに渡す情報をユーザに明示する機能を持たせる	他のサービスに渡した情報が何かを確認する				○		○	○
11		Webアプリケーションファイアウォール	Webアプリケーションファイアウォールを利用できるようにする	Webアプリケーションファイアウォールを利用できるようにする	Webアプリケーションファイアウォールが利用できることを確認する						○	○

図1-1 チェックリスト項目図

①	No	項目番号
②	大項目	IoTセキュリティへの具体的な対策を検討する上で必要となる39の小項目に対して、どの分野の機能において、セキュリティを考慮するかという観点で次の8つの大分類に分けています。 (1)ユーザ管理(2)ソフトウエア管理(3)セキュリティ管理(4)アクセス制御(5)不正な接続(6)暗号化(7)システム設定(8)通知
③	小項目	
④	本項目の目的	満たすべきセキュリティの要件を記載しています
⑤	開発する際に気を付けること	IoTシステムの開発側、利用側にて確認する項目は異なるため、別の項目を用意しており、開発者・利用者双方が同じ言葉にて、同じ項目を確認することにより、共通の認識を持ちやすくしています。
⑥	利用する際に気を付けること	
⑦	チェック欄	チェック欄の他に理由欄も設けており、単純に機能の有無の評価を行うのではなく、評価する機能に対して、どのような対応を行っていくのか検討を行えるようにしています。
⑧	理由	
⑨	関連するコンポーネント	IoTシステムを構成する機能のうち、どの機能についての確認項目であるかを示している。 それぞれ、「(S)Sensor」、「(A)Aggregator」、「(E)e-Utility」、「(D)Decision Trigger」、「(C)Communication Channel」を示しています。

図 1-2 チェックリストの項目

IoTシステムの機能ごとの分類

先に述べたようにIoTセキュリティチェックリストでは、IoTシステムのセキュリティを評価する際に、IoTを構成する機能ごとに評価を行うため、まず評価対象を機能ごとに分類を行う必要があります。

IoTチェックリストでは、NIST SP 800[*5]-183を参考に、IoTは次の5つの機能によって構成されているとしています。5つの機能の簡単な説明は次のとおりです。

- Sensor:
  温度、加速度、重量、音、位置などを測定するための機能
- Aggregator:
  センサーからのデータを集約して処理をする機能
- Communication Channel:
  データの送受信を行うための通信路
- eUtility:
  サービスを提供する／受ける機能
- Decision Trigger:
  データの加工や計算をするための機能

これらの分類について、IoTチェックリストを使ってセキュリティチェックを行いたいIoT製品に当てはめます。たとえば、一例としてネットワークカメラの構成に対して機能ごとの分類を行ったものは次のとおりです。(図2)

IoTセキュリティチェックリストでは図のように、対象のIoT製品の各機能に対して分類を行い、IoTのシステム全体を評価したい場合は、それぞれの機能ごとに該当する確認項目を確認し、そのセキュリティの機能が、5つの分類のいずれかの部分で実装されているかどうかの検討をする形になります。

また、システム全体ではなく、デバイス部分のみを評価したい場合においても、そのデバイスがどの部分に該当するのかを確認し、対象の項目をチェックすることで、評価対象に不足しているセキュリティの機能について、許容するのか、他のセキュリティアプライアンスにて補うのか等の検討を行う事ができます。

3-3 項目の選定基準について

IoTセキュリティチェックリストの小項目は、IoT以外にIT向けのセキュリティ評価資料を参考にしています。これはIoT製品が既存ITの技術を利用しながら構成されているケースがあり、IT向けのセキュリティ評価が参考になると考えるためです。例えば、IoT製品に含まれるeUtilityなどに実装されるインターフェースは、Webアプリケーションや、APIによるアクセスなどの通信プロトコルがWebとよく似た実装となっている、ということがあります。IoTセキュリティチェックリストでは、数多くあるIoT/IT向けのセキュリティ評価資料の中でも、特に具体的なセキュリティ評価の方法が記載されている次の文章を主な参考としています。

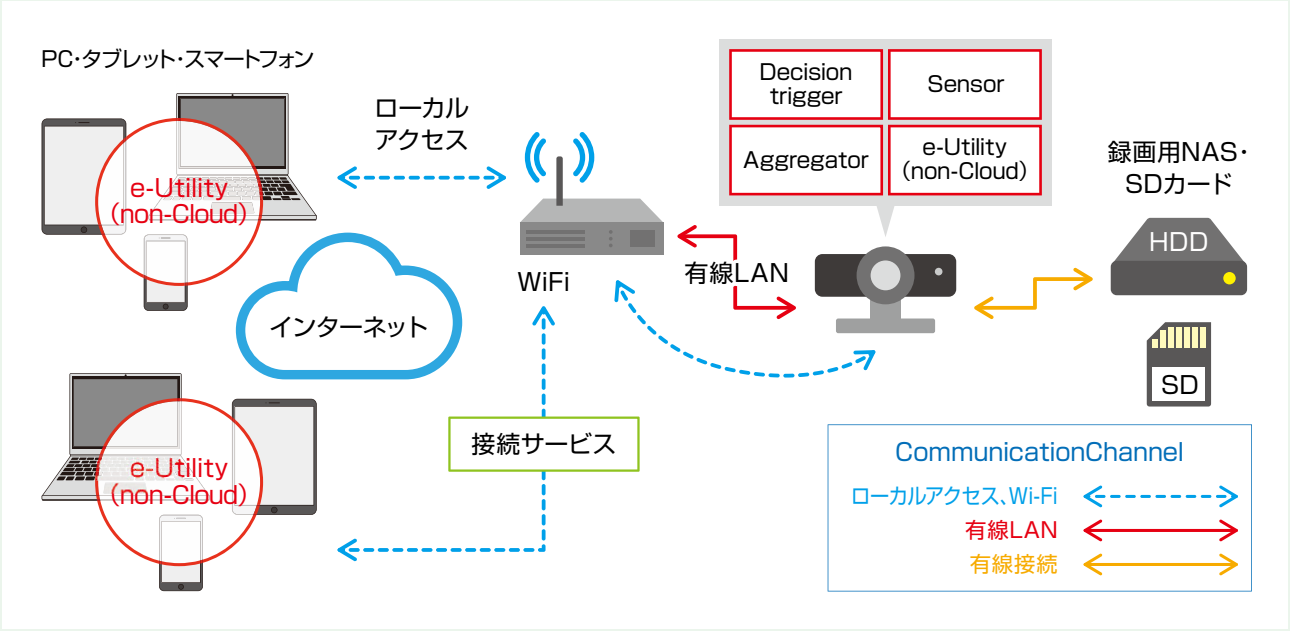


図 2 例：ネットワークカメラの構成に対して機能ごとの分類

*5: NIST SP 800シリーズは米国国立標準技術研究所が発行しているコンピュータセキュリティに関するレポート

OWASP IoT Security Guidance
セキュアなソフトウェア開発を促進する技術に関する情報共有や、普及啓発などを目的とし、世界中より専門家が集まるオープンソース・ソフトウェアコミュニティOWASP(Open Web Application Security Project)が公開しているIoTセキュリティに関するチェック事項

The Penetration Testing Execution Standard (PTES)
PTESは、Penetration Testing Execution Standard Groupにより、ペネトレーションテスト時に、テスターが行う作業の要件がまとめられた文章であります。バージョン1では標準のコアとなる要素をまとめ、1年以上にわたって利用実績のある文書です。

上記の2つの資料から、共通する項目を基準項目として検討を行っています。これらの資料は、ITおよびIoTのセキュリティを考える上で、特に基本的な項目がまとめられており、項目の中にはそれぞれの資料の中で共通しているものが多く存在します。

そのため、これら共通項目は注目度が高く優先して実施することが望まれるのではないかと考えています。

しかし、本チェックリストでは、これらの項目はすべてが満たされなければならない要求要件ではないと考えています。つまり、注目度が高いことから、優先的に対策を講じるなどの議論や検討が必要とされている推奨項目であると考えています。なぜならば、IoT製品やシステムによっては、その他に求められる規格などの要件や、項目に重み付けがあるなど考えられるはずであり、本文書のチェックリストを適用する際には、それらの他の要件とも併せて検討することが望ましいと考えるからです。また、問題やデバイスによっては、セキュリティ要件による解決ではなく、IoT化する機器側にもともとついているセーフティに対する要件によって解決を図ることが優先できる場合も考えられます。そのため、IoTセキュリティチェックリストではセキュリティに対する基本的な検討項目をまとめています。これを参考にしつつ、評価対象としているIoT製品の特性を考慮したうえで、セキュリティを考えていっていただけると幸いです。

おわりに

本稿では次の二点についてご紹介しました。

IoTシステムにかかわる脅威や問題について

日本を含め、世界中でMiraiをはじめとしたさまざまなマルウエアなどによるサイバー攻撃が継続して日々、行われております。それらの攻撃はIoT製品の設定の甘さや脆弱性を悪用したものであり、それらに対応するため、適切な設定や脆弱性対応の呼びかけや、そもそもセキュアな製品の開発が呼びかけられています。しかし、それ以外にもIoTにまつわる問題として、IoTのセキュリティの範囲の理解や開発者と利用者の認識の違いといった問題が存在します。

IoTセキュリティチェックリストについて

JPCERT/CCでは、IoTシステムにかかわる脅威や問題について、解決するための助けとなるようべく、IoTセキュリティチェックリストを作成しました。

このIoTセキュリティチェックリストには次のようなポイントがございます。

- IT、IoTにおいて、共通で求められるセキュリティに関する項目を羅列している
- IoTシステムの機能ごとに確認するポイントを分けている
- 開発者・利用者ごとの確認内容を設定している

尚、本稿にて紹介しましたIoTセキュリティチェックリストをご利用になりたい方はJPCERT/CC早期警戒グループ (ww-info@jpcert.or.jp) までご連絡ください。

今回、本稿にてご紹介した内容が皆様の活動の参考になれば幸いです。

参考資料

- “Mirai Botnet”. https://www.cyber.nj.gov/threat-profiles/botnet-variants/mirai-botnet
- “Multi-exploit IoT/Linux Botnets Mirai and Gafgyt Target Apache Struts, SonicWall”. https://researchcenter.paloaltonetworks.com/2018/09/unit42-multi-exploit-iotlinux-botnets-mirai-gafgyt-target-apache-struts-sonicwall/
- “KrebsOnSecurity Hit With Record DDoS”. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/
- “MMD-0056-2016 - Linux/Mirai, how an old ELF malcode is recycled..”. http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html
- “Source Code for IoT Botnet ‘Mirai’ Released”. https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/
- “IoT Security Guidance”. https://www.owasp.org/index.php/IoT_Security_Guidance
- “The Penetration Testing Execution Standard”. http://www.pentest-standard.org/index.php/Main_Page

本記事の内容に関するご質問、お問合せは、**ベリサーブマーケティング部** Email:verinavi@veriserve.co.jp までお寄せください。

山口 鉄平氏
やまぐち てっぺい

アジャイルコーチ/ソフトウェアコンサルタント。
モノづくりを試行錯誤する中でアジャイル開発やテスト技術に出会い、それらの研鑽や普及を行っている。
現在の主な活動は、テスト自動化研究会 お世話係、一般社団法人アジャイルチームを支える会 監事など。
JaSST'16 Hokkaidoでの基調講演などの講演や共訳書に『Fearless Change(2014丸善出版刊)』がある。

アジャイルなソフトウェア開発におけるテスティングの実際

はじめに

ここ数年、国内でもScrum[*1]などのアジャイルなソフトウェア開発の話を見聞きする機会が増えてきています。そのようなアジャイルなソフトウェア開発の中では、ウォーターフォールをはじめとする従来型のテスティングではいくつかの課題が発生します。

本稿では、それらの課題ならびに、それらの解決方法やアジャイルな開発でのテスティング事例を紹介します。なお、以降での"開発"とはアジャイルなソフトウェア開発を示すものとします。また、その"開発"の具体的な方法はさまざまなものが存在するため、多くの方法が有している下記の特徴を持つソフトウェア開発を本稿での"開発"とします。

- 反復的な開発とその反復ごとに検証可能なソフトウェアを提供する
- 検証可能の最小単位として、優先順位がつけられたストーリーと呼ばれるものごとに開発を進める
- 提供するソフトウェアに責任を持つチームで開発を行う

1. アジャイルな開発でのテスティングの課題

【課題1】アジャイルな開発への従来型テスティング適用による課題

アジャイルな開発においては、反復ごとの検証に基づきソフトウェア全体が変化していくため、ソフトウェアの全体を見通してテスト計画・テスト分析・テスト設計・テスト実装・テスト実行といったテストプロセスを1回だけ実行する従来型のテスティングは適用できません。(図1)

*1: Ken Schwaber、Jeff Sutherlandにより開発されたアジャイルなソフトウェア開発手法の一つ。
複雑で変化の激しい問題に対応するためのフレームワークとして定義されている。https://www.scrumguides.org

また、反復ごとに前記のテストプロセスを実施する場合、優先順位の高いストーリーに不具合を発見した際に修正が間に合わず、反復終了後にソフトウェアを提供できない状況が多く見られます。(図2)望む形としては、優先順位が高いストーリーをいち早くリリースするために、不具合修正も含めて優先順位の高いストーリーに時間を使うべきですが、従来型のテスティングではテスト実行が反復後半になりやすく、不具合の発見が遅れがちです。

【課題2】SETを置くことによる課題

アジャイルな開発や、より早いソフトウェアのリリースに向け、国内でもテストコードを書く文化が広がってきているように感じます。それに伴い、最近、テストコードやテスト環境を整備する役割であるSET(Software Engineer in Test)がWEB企業を中心に設けられるようになってきています。しかし、その中にはSETを置くことによる課題を認識せずSETを設けていることがしばしば見られます。その課題として、大きく次の3つが考えられます。

①役割の設置による責任の曖昧化
②テストに向けたコミュニケーションコストの大きさ
③プログラミングとテスティングを、時間をおいて実施することの非効率性

SETにテストコードの作成などを責任として持たせた場合、相対的にSET以外のメンバのテスティングに対する責任が低下しやすくなります。(①)

また、開発に関わる人が増えれば増えるだけ、テストに向けた伝言ゲームによるコミュニケーションコストが増大します。加えて、伝言ゲームによる不具合発生を防ぐためにコミュニケーションコストが増大します。(②)

最後に、SETを置くことによりプログラミングとそれを対象としたテストの作成・実施に時間的間隔が生まれやすくなる傾向があります。プログラミングとテスティングを、時間をおいて実施することの非効率性は、論文 “A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last?”などで示されています。(③)前述の論文はTDD(テスト駆動開発)の順序・

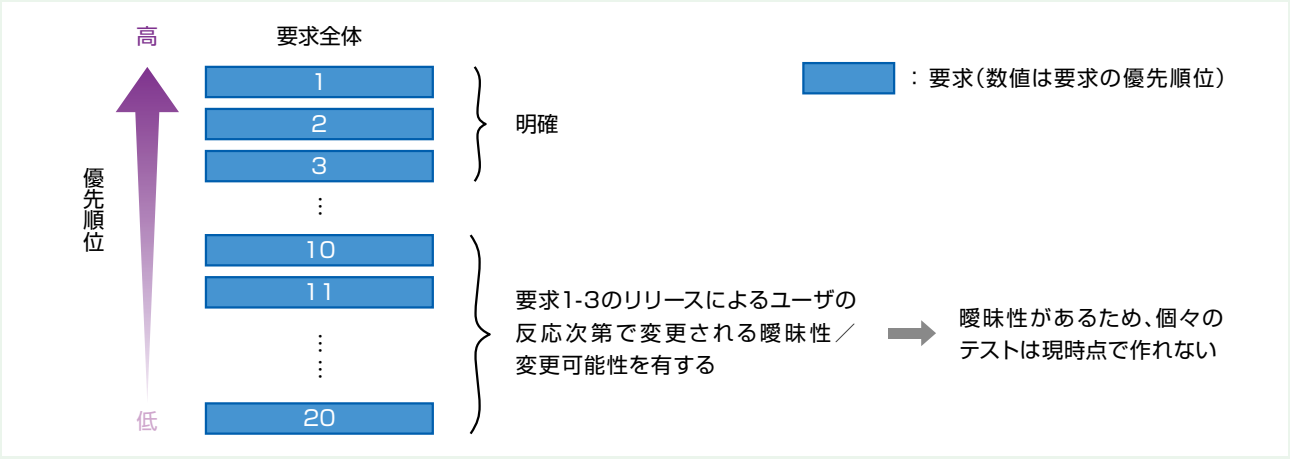


図 1 要求が変化するため従来型のテスティングの適用ができない

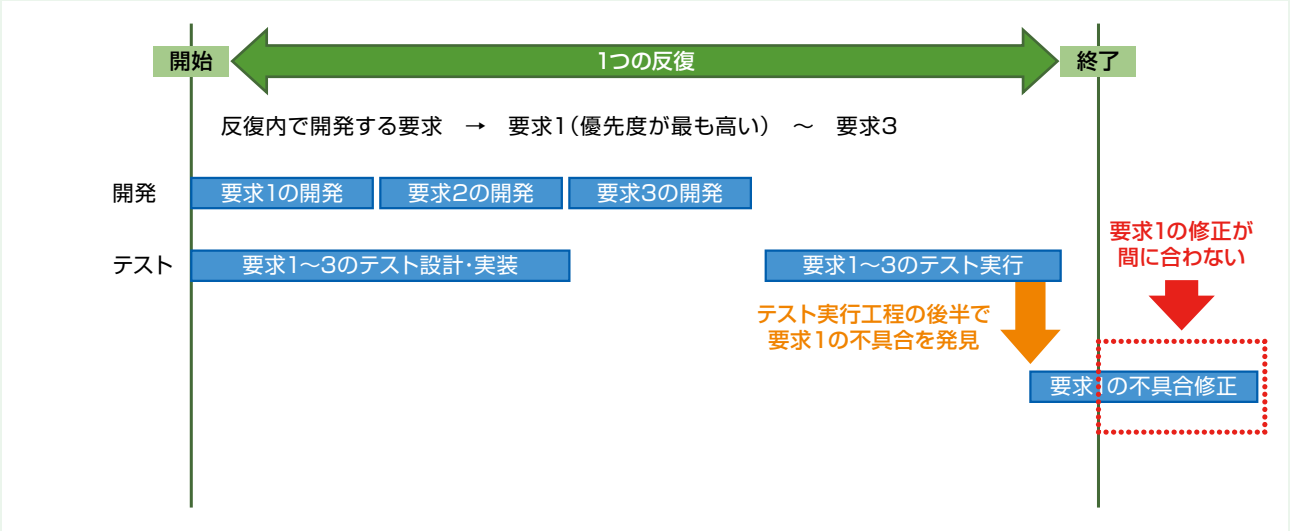


図 2 優先順位の高いストーリーの修正が間に合わない

粒度・均一性・リファクタリングに関する調査です。その調査では、TDDのプログラミングとテストコード作成・実行それぞれの作業を短くし、かつ一定のサイクルで交互に実施することが、品質と開発効率の改善には効果があると述べています。

これらのため、筆者としては安易にSETを設けることには課題があると感じています。実際、10年程度位前のMicrosoftやGoogleなどでSETは存在しましたが、現在そのような役割はなくなっています。

2. アジャイルな開発でのテスティングの変化

2-1 解決策としてのアジャイルテスティング

上記のように、アジャイルな開発において従来行っているテスティングをそのまま適用すると課題が発生することがあります。そのような課題に対しては、書籍「実践アジャイルテスト(原著:Agile Testing)」や「ISTQB® Agile Tester Extension Syllabus」に書かれていることがヒントになります。

書籍「Agile Testing」の著者であるLisa CrispinとJanet Gregoryは、「Agile Testing」やその続編である書籍「More Agile Testing」内および最近まで"アジャイルテスティング"と言う言葉に対して定義を与えていませんでした。そのため、彼女らが考える"アジャイルテスティング"が端的に表現されていませんでした。そこで、彼女らは2017年にブログ上の記事「Our Definition of “Agile Testing”」にて定義を発表しました。その記事をkz_suzukiさんが翻訳してくださったブログ上の記事「【翻訳】「アジャイルテスト」、わたしたちの定義」から"アジャイルテスティング"の定義を下記に引用します。

アジャイルテストの定義(Lisa Crispin、Janet Gregory)
開発の始めからデリバリーまでに行われる協働的なテストのプラクティスで、顧客にビジネス価値をもたらす高品質なプロダクトの、高頻度なデリバリーを支える。テストの活動は、欠陥を見つけることではなく抑えることにフォーカスするものであり、「品質に対するチーム全体の責任」という考え方を強め、支えるためにある。
アジャイルテストは、以下のようなテストの活動を含む(が、これに限られない)。
具体的な例を挙げて開発を導く。アイデアや仮定をテストするために質問を投げかける。テストを自動化する。探索的テストを行う。性能、信頼性、セキュリティといった品質特性のためにテストする。

上記定義や書籍ではテストエンジニアの視点からの話も多いのですが、書籍およびJanetが行うAgile Testingの研修では、アジャイルな開発でのテスティングは"Whole Team Approach"であることが強調されます。その"Whole Team Approach"とは何かを筆者なりにまとめると下記の3点と考えています。なお、ここでの"品質"とは、不具合などプロダクトに関することだけでなく、いち早く顧客が利用できるなどビジネス価値を含むような広範囲での"品質"を指すため、あえて"質"と表現しています。

①質はビジネス、開発、テストなどチーム全体の責任である
②テストエンジニアだけでなくプログラマを含むチーム全員が、テスティングに開発初期から関わり続ける。
③テストエンジニアはテスティングや自動化だけではなく、完成の定義や要求の明確化に向けた質問をする

加えて、SETの役割をなくしたMicrosoftでは、クラウド時代の品質に向けて変化させたこととして、HP上のドキュメント「Evolving Test Practices at Microsoft」にて下記の3点を紹介しています。

1. We redefined quality ownership, we fixed the accountability.
2. We understood that in order to ship frequently, Master branch must always remain as healthy as the release branch. We defined a core principle - Master is always shippable. The principle touches everything - source code management, code practices, build etc. From testing perspective, we pushed two things: shift-left testing (i.e. greater emphasis on unit testing) and eliminating flaky tests.
3. We also understood there is no place like production. This is the shift-right part of the strategy. it’s a set of practices about both safeguarding the production as well as ensuring quality in production.

(筆者翻訳による要旨)
1. 我々は質のオーナーシップを再定義し、その説明責任を修正した。
2. 頻繁にソフトウェアを提供するために、マスターブランチはリリースブランチと同じくらい健全な状態を保つ必要があることを、我々は理解した。我々はコア原則を定義した - マスターブランチは常に提供可能な状態にする。その原則はすべてに関わる - ソースコード管理、コードプラクティス、ビルドなど。テスティングの観点からは2つのことを推し進めた、シフトレフトテスト(すなわち、ユニットテストの重視)と不安定なテストの排除である。
3. 我々は本番環境のような場所がないことも理解した。これは、テスト戦略のシフトライトの部分である。本番環境を保護し、本番環境での質を保証することの両方のプラクティスのセットである。

Microsoftが変化させたことの1つ目でも同様のことを紹介するように、アジャイルな開発でのテスティングでは、役割ごとの責任ではなく、ソフトウェア開発に関わる全員が質に対して責任を持つことが重要となります。そして、テスティングが得意なチームメンバは、不具合やソフトウェアの状態を含むさまざまなことをより早くフィードバックするために、開発初期から関わることが求められます。

そこでは、一般的なテスティングだけでなくビジネスや要求への質問によるフィードバックを行うことで、アジャイルな開発でのテスティングを実現します。また、プログラミングが得意なチームメンバもプログラミングだけでなく、質問によるフィードバックやテスティング、それらの自動化などを行うことで、アジャイルな開発でのテスティングを実現します。このようにアジャイルな開発でのテスティングは、チーム全員により実施されます。

2-2 「実践アジャイルテスト」で紹介されるテスティングアクティビティ

前節ではアジャイルテスティングの概要について紹介しました。アジャイルテスティングでも、テスティングとしての「計画・設計・実装・実行・報告」を担う行為がなくなることはありません。そこで本節では、読者の方にも馴染み深い、テスティングとしての「計画・設計・実装・実行・報告」といったテスティングアクティビティの視点で、さらにアジャイルテスティングについて紹介を進めます。特に、書籍「実践アジャイルテスト」の事例を元に、それらを示します。なお、この書籍はアジャイル開発へ関わることになったテストエンジニアであるLisa CrispinとJanet Gregoryが、どのように立ち居振る舞うことで開発に貢献できるかを自身らの体験を通じてまとめた書籍であり、事例も掲載しているため具体的にわかりやすいものとなっています。

以降、読者の方に馴染み深い、テスティングとしての「計画・設計・実装・実行・報告」といったテスティングアクティビティが、アジャイルな開発の中のどのような活動として実施されるかを紹介します。図3はアジャイルな開発の全体像を抽象的に表現したものです。従来のテスティングアクティビティに相当する活動はさまざまな箇所で行われますが、その中でも本稿では①~⑧の数字の箇所で行われる活動を紹介します。

2-2-1 テスト計画およびテスト設計

テスト計画やテスト設計のアクティビティは、アジャイルな開発では下記の活動などとして行われます。

①反復開始前のストーリー作成および認識合わせ時に、そのストーリーのテストのスコープ検討など、テスト計画を行います。それと共に、どのようなテストができればストーリーが完了となるか、新たなストーリーがソフトウェア全体へどのように影響を与えるか検討して、テスト設計を行います。
②反復開始前のタスク作成および認識合わせ時に、どのようなテストができればタスクが完了となるかテスト設計を行います。
③プログラムの細かなメソッドなど作成時に、どのようなテストができればそのメソッドが持つべき責務が確認できるかテスト設計を行います。

2-2-2 テスト実装および実行、報告

テスト実装やテスト実行、報告のアクティビティは、アジャイルな開発では下記の活動などとして行われます。

④反復開始前のストーリー作成および認識合わせ時に、質問を通じて、そのストーリー自体すなわち要求のテストを実施します。
○ ストーリーが実現した際の例を確認し、要求を確認もしくは明確化します。

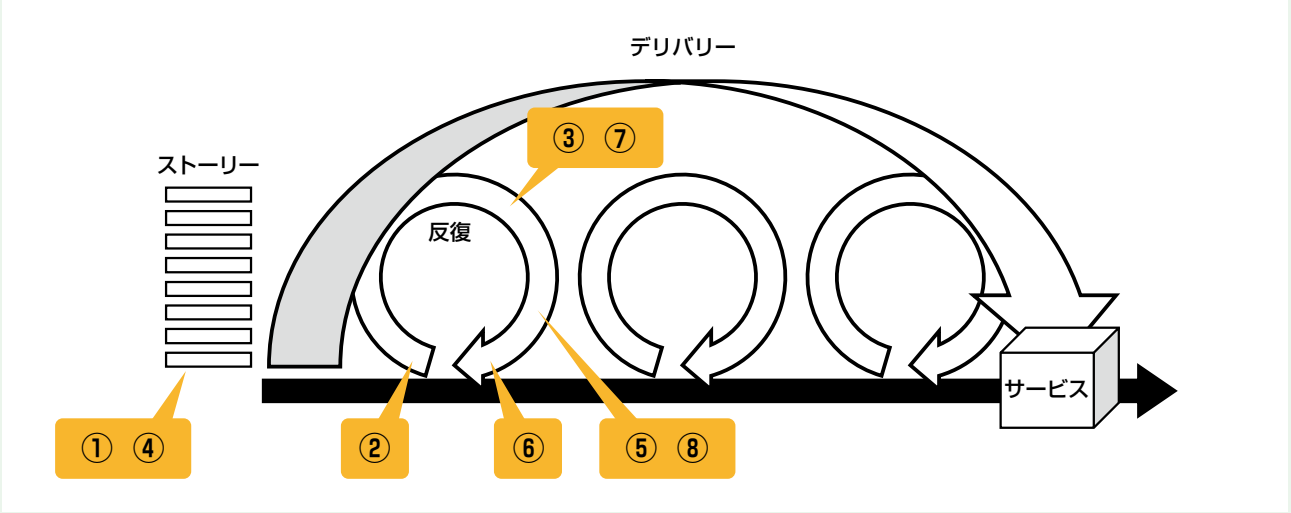


図 3 アジャイルな開発でのテスティングアクティビティ

⑤反復内の各タスクやストーリーの完了を確認するためのテストを実施します。
⑥反復内で、可能な限り顧客を開発の輪の中に入れ、顧客に何度も早期にレビューしてもらうようにします。
⑦プログラムの細かなメソッドなどの作成時に、責務を確認するためのテストケースもしくはテストコードの実装・実行をします。
　○ シンプルなテストで開発を促進し、そのテストをパスしたら、さらに複雑なテストで開発を促進します。
⑧見つけた不具合をチームメンバに報告する。不具合の内容をチームで話し合い、即時に修正するか、新たなストーリー・タスクを追加するか決定します。

複数の反復で開発したソフトウェアに対して、それら期間の最後に一時的に開発を停止しテスティングを行うことで品質の確定などをするエンドゲームと呼ばれる方法も書籍中では紹介されています。しかし、この方法は書籍中では"必要ならば"と語られていますし、筆者の経験からもエンドゲームを常時用いることは反復開発の意味を損なうため推奨しません。

3. アジャイルな開発でのテスティング事例

ここでは、実際のアジャイルな開発でのテスティングの事例を紹介します。

3-1 全員プログラマのチームでの事例

専門のテストエンジニアが不在であり、全員がプログラマであるチームでの事例を紹介します。なお、このチームのプロフィールとしては下記となります。

- 開発プロセス:
  Scrum
- 反復期間:
  1週間
- メンバ構成:
  6名（全員プログラマ、うち一名はプロダクトの企画責務を有する役割を兼務）
- 開発対象:
  リリース済みのWEBサービス。一定の利用者がおり、より使いやすく利用者を増やすための開発を継続している。

このチームでは、テスト計画・設計のテスティングアクティビティとして下記のようなことを行っています。（表1-1）
また、このチームでは、テスト実装・実行・報告のテスティングアクティビティとして下記のようなことを行っています。（表1-2）
この事例は次節の"プロダクトマネージャ中心でテスティングを進める事例"と比較して、チームメンバによるテスティングが多いものとなっています。そのため、プロダクトの質が継続的に安定したモノになりやすい反面、チームメンバのプロダクトへの理解にばらつきがある場合、プロダクトの質が安定しないリスクが高くなっています。

実施シーン	実施内容
反復開始時のその反復内での業務計画および、次回以降の反復で行うストーリーの見直し・計画時	各ストーリーやプログラミングタスクに対して、何をしたらそれらが確認できるのかチームで議論し、それらをストーリーやタスクの完了条件として記載する。 ※ストーリー単位の完了条件は大きなテストなどで構成され、プログラミングタスク単位の完了条件は確認するパラメータなどを含むタスクの大きさに応じた小さなテストなどで構成される。
	ストーリーに関する議論を踏まえ、テストにかける時間を含む形で、その反復で実現するストーリーを決定する。
	テストを自動テストとするかを決定する。
プログラミングタスク実施時	クラスやメソッドなどが持つ責務を確認するためのパラメータや条件を決定する。

表1-1 テスト計画・設計のテスティングアクティビティ

実施シーン	実施内容
反復開始時のその反復内での業務計画および、次回以降の反復で行うストーリーの見直し・計画時	ストーリーやタスクを作成する際に、チームでストーリーやタスクに対して質問し合うことにより、ストーリーやタスクで要求を確認する。
タスク実施時	完了条件として決めた確認事項を実施する。
	クラスやメソッドなどが持つ責務を確認するためのテストケースおよびテストコードの実装・実行を行う。
	テストケース／テストコードを含めてコードレビューを行う。
	自動テストの実装・実行を行う。
	クラスやメソッドなどが持つ責務を確認するためのテストケースおよびテストコードの実装・実行を行う。

表1-2 テスト実装・実行・報告のテスティングアクティビティ

3-2 プロダクトマネージャ中心でテスティングを進める事例

プロダクトマネージャが中心となってテスティングを実施するチームでの事例を紹介します。なお、このチームのプロフィールとしては下記となります。

- 開発プロセス:
  反復ごとのミーティングはScrum同様だが、ロールはスクラムマスターが不在等、独自の開発プロセス
- 反復期間:
  1週間
- メンバ構成:
  プロダクトマネージャ1名、プログラマが4名、デザイナが1名
- 開発対象:
  リリース済みのWEBサービス。ソフトウェアとしてはすでにリリースし、これから利用者を増やすための開発を継続している。

このチームでも、テスティングアクティビティの多くは前節の"全員プログラマのチームでの事例"と同様です。異なる部分のみを下記に紹介します。

前の事例と異なるテスト計画・設計のテスティングアクティビティとしては、下記のようなことを行っています。(表2-1)

また、前の事例と異なるテスト実装・実行・報告のテスティングアクティビティとして下記のようなことを行っています。(表2-2)

この事例は前節の"全員プログラマのチームでの事例"と比較して、プロダクトマネージャが担うテスティングが多いものとなっています。そのため、プロダクトマネージャが持つソフトウェアに対するイメージが効率的に反映されやすい反面、プロダクトマネージャへの負荷集中の発生や継続的にソフトウェアを提供していくリスクが高くなっています。

おわりに

本稿では、アジャイルなソフトウェア開発の中でのテスティングの課題ならびにそれらの解決の方法、事例を紹介しました。

アジャイルなソフトウェア開発でのテスティングは、役割ごとに責任を負うのではなく、ソフトウェア開発に関わるすべての人で、いかに早くフィードバックを返すか考え、実施していくことが基本となります。そのために、テスティングが得意なチームメンバは開発初期から関わり、一般的なテスティングだけでなく要求の明確化に関わると共に、プログラミングが得意なチームメンバはプログラミングだけでなく、テスティングやそれらの自動化などを行うものとなります。

本稿が、読者の開発に役立てば幸いです。

実施シーン	実施内容
反復開始時のその反復内での業務計画および、次回以降の反復で行うストーリーの見直し・計画時	プロダクトマネージャが反復ごとのリリース前に、ソフトウェア全体に対してどのような確認をするのか、どの程度期間をかけるのかをリリースタスクとして作成する。

表2-1 テスト計画・設計のテスティングアクティビティ

実施シーン	実施内容
リリースタスク実施時	プロダクトマネージャがストーリーの完了条件として決めた確認事項の実施

表2-2 テスト実装・実行・報告のテスティングアクティビティ

参考文献

- Lisa,Crispin.; Janet,Gregory. 実践アジャイルテスト
  -- テスターとアジャイルチームのための実践ガイド. 榊原 彰, 増田 聡, 山腰 直樹, 石橋 正章 訳. 翔泳社, 2009, 560p.
- “The Agile Tester Extension is based on the ISTQB® Agile Tester Extension Syllabus”. https://www.istqb.org/downloads/syllabi/agile-tester-extension-syllabus.html
- Lisa,Crispin.; Janet,Gregory. “Our Definition of “Agile Testing””. https://agiletester.ca/definition-agile-testing/
- kz_suzuki. “【翻訳】「アジャイルテスト」、わたしたちの定義”.http://www.kzsuzuki.com/entry/2017/07/06/234332
- Munil,Shah. “Evolving Test Practices at Microsoft”. https://docs.microsoft.com/en-us/azure/devops/learn/devops-at-microsoft/evolving-test-practices-microsoft
- James,A.W.; Jason,Arbon.; Jeff,Carollo. テストから見えてくる グーグルのソフトウェア開発. 長尾 高弘 訳. 日経BP社, 2013, 432p.
- Davide,Fucci. et al. A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last?. IEEE Transactions on Software Engineering.2017, vol. 43, no. 7, p. 597-614.

本記事の内容に関するご質問、お問合せは、ベリサーブマーケティング部 Email:verinavi@veriserve.co.jp までお寄せください。

連載【前編】

デジタルイノベーションがもたらす創造的破壊とは

～ソフトウェア大革命に備えよう～

大原 茂之氏
おおはら しげゆき

現在（株）オプテック代表取締役会長、工学博士
東海大学名誉教授、九州工業大学客員教授、一般社団法人スキルマネージメント協会理事長

はじめに

デジタルは言い古された言葉ですが、デジタルイノベーションという用語からはどこか新しさと期待感が生まれてきます。

AI、IoT、ロボットなど先端的な技術の導入意欲が、製造、サービス、医療、金融などさまざまなドメイン（事業領域）でデジタルイノベーションを起こしつつあります。

ただし、この言葉は標準的に定義されている訳ではありません。

前編ではソフトウェア技術そのものがデジタルイノベーションによって受けるインパクトについて探っていきます。

1. イノベーションとは何か、技術とは何か

1-1.イノベーション

デジタルイノベーションを理解する前に、そもそもイノベーションとは何かそしてどのように拡大していくかについて理解しておきましょう。

一般社団法人スキルマネージメント協会（SMA）では、イノベーションを起こしていくスキルとイノベーションを起こす環境について研究を進めています。図1はそうしたSMAの活動から生まれたモデルで、イノベーション・イノベータモデル（I^2モデル）とよんでいます。このモデルは、シュンペータが提唱したイノベーションの特性と最新のキャズム理論をそれぞれモデル化して合成したものです。図1の内側の幅広の円（提供側）はシュンペータのイノベーションのモデルで、外側の幅広の半円（購入側）はキャズム理論のモデルです。これらの円の半径はイノベーションによる経済規模に相当するとご理解下さい。巨大な半径になると、経済構造を根底から覆す創造的破壊型のイノベーションになります。例外はあるとしても、このモデルは利用者のニーズが先導してイノベーションを起こすのではなく、内側からの製品やサービスの提供がイノベーションを先導し、それをイノベータからラガードまでに分類される利用者が市場を形成していくことを示しています。提供側と利用者側のサイクルが循環してイノベーションを加速していくのです。

一方、製品やサービスの中の部分的あるいは日常業務の一部を改善改良することは、小さな半径のイノベーションとなり持続的イノベーションなどとよばれます。このように、イノベーションは数十年あるいは数百年に一度という大規模のものから、日常茶飯事起きる小規模のものまでさまざま存在します。

1-2.キャズム

前節で述べたように、市場においては新技術による新製品や新サービス（以下、新製品）に飛びつく人たちをイノベータとよび2.5%程度存在していると考えられます。イノベータの先にはクラック（割れ目）が邪魔をしていますが、これを越えると13.5%存在すると考えられるアーリーアダプタの集団がいます。さらにアーリーアダプタの先にキャズムという大きな溝が存在しますが、これを越えるとアーリーマジョリティとレイトマジョリティからなる68%ものボリュームゾーンが登場します。新製品の提供側がイノベーションを成功させるには、これらのクラックやキャズムを乗り越えて利用者側を巻き込んでいく戦略が必要なのです。

基本的には次のようになります。

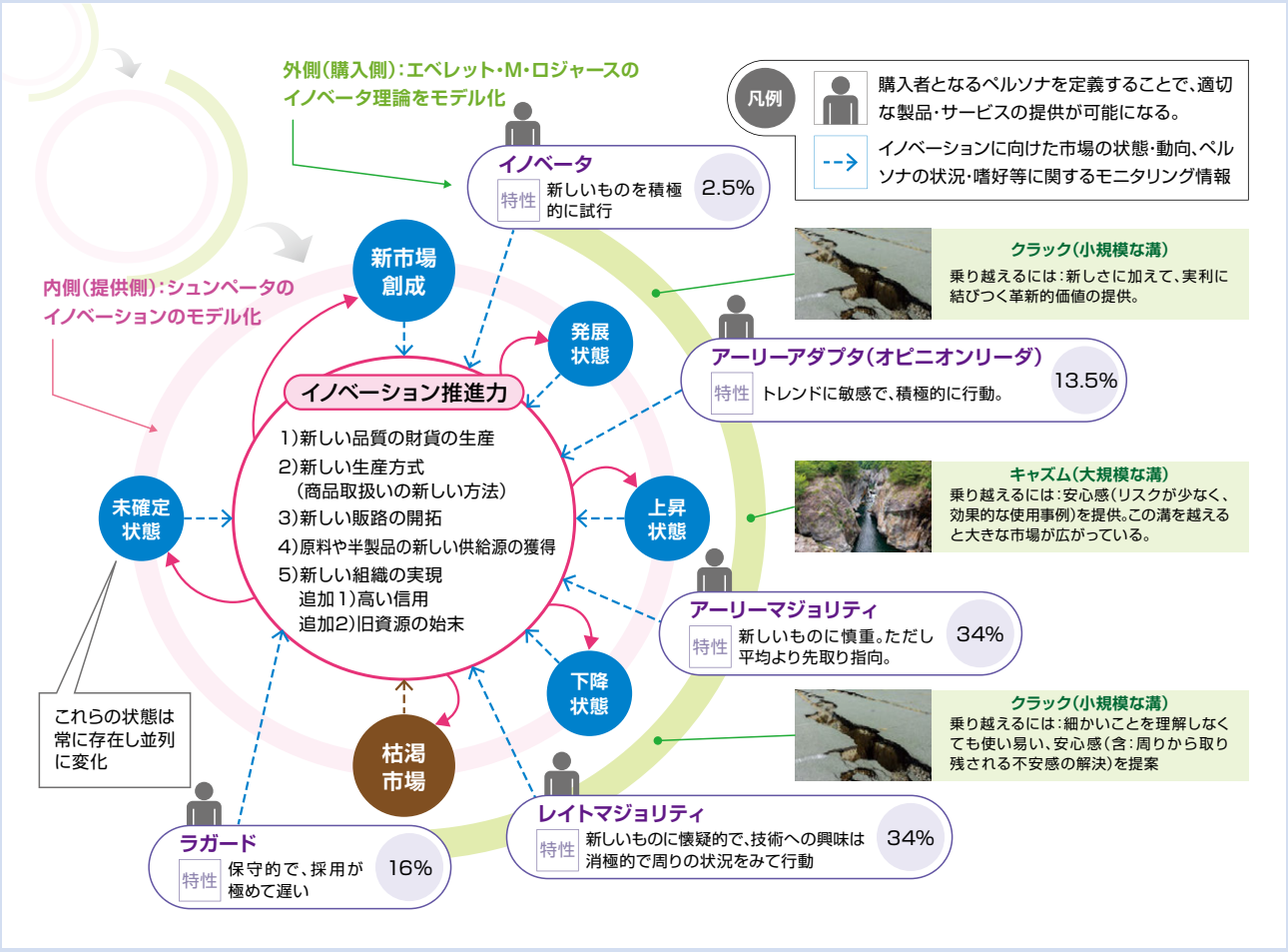


図1 イノベーションの構造

①新製品とそのコンセプトからイノベータを探索
②イノベータの振る舞いの観察・意見の収集などから、アーリーアダプタへの障害となるクラック制覇の方策を立案
③アーリーアダプタの振る舞いの観察・意見の収集などから、アーリーマジョリティへの障害となるキャズム制覇の方策を立案
④アーリーマジョリティの振る舞いの観察・意見の収集などから、レイトマジョリティへの障害となるクラック制覇の方策を立案

1-3.技術とスキル

イノベーションを成功させるには、図1のイノベーションの中心にあるイノベーション推進力を強化する必要があります。この推進力は技術と技術を使いこなす人材のスキル(習熟度)に依存します。

【技術】
技術とは、経済性、新規性、再現性、保守性を満足しつつ、要求に対応する結果を得るまでの知識化された工程(プロセス)のことです。技術は、工程を表わす設計図やプログラムといった文書、あるいは工程を内包する物理的な製品として作られます。
知識としてさまざまに表現可能な技術は、一人から多数へ伝達可能という特性をもっています。

【スキル】
スキルとは技術を使いこなす能力です。この能力は長年に渡って獲得する経験や暗黙知など個人の中に醸成されるもので、その能力を明確に数値化できないアナログ的なものなのです。
知識化できないスキルは、人から人への体系的な伝達が困難なのです。

図2(P.16)は技術とスキルの関係を示したもので、技術とスキルが充足している場合は目的を達成できますが、どちらか一方でも不足すると、目的は達成できなくなります。

技術は知識であるため工学的に実現および再現可能ですが、経験値や暗黙知を包含するスキルは工学的に実現可能だろうかという疑問が生じます。

これまでは不可能でしたが、AIの登場が間接的に可能にしつつあるのです。経験値や暗黙知というアナログ領域を、AIというデジタル技術が吸収できるようになりつつあります。

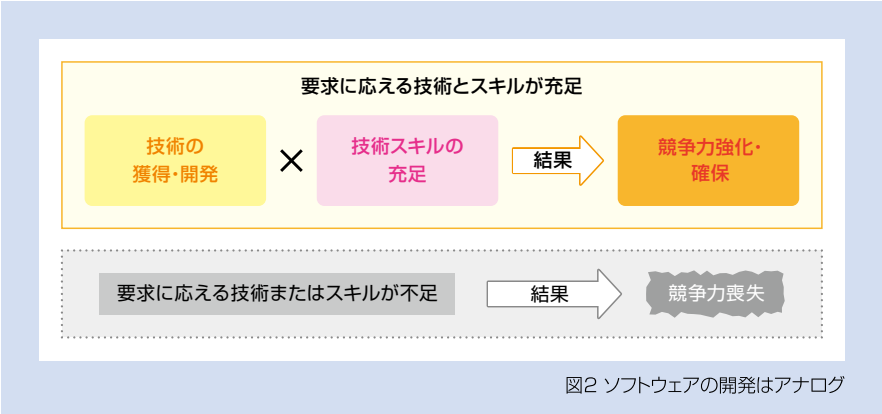


図2 ソフトウェアの開発はアナログ

2. ソフトウェア開発はアナログ!

2-1.工業製品

自動車など量産を前提とする工業製品の場合、開発した製品をベースに高品質化と低コスト化を徹底的に追求する量産設計を行います。量産においてはネジや電子部品など多種多様な部品を製造する川上産業から始まり、製品を組み立てる川下産業に至るまでのサプライチェーンが構築されます。

高品質・低コスト・量産を優先することから、使用する部品は旧部品に縛られることなく、新しい部品やモジュールを使用する設計が求められます。

例えば、電子部品の変遷でいえば、真空管からトランジスタへ、次にIC(集積回路)へ、さらにLSI(大規模集積回路)やVLSI(超大規模集積回路)へ、今ではその集積度もスイッチング速度も飛躍的に進歩しコストダウンも加速しています。

製品の方も、強力な新部品や製造技術が登場すると、旧製品に引きずられることなく最先端の技術を使用し、新たな製品を設計・開発・製造するのです。高品質化や低コスト化の追求により、デジタル化は必然的な流れとなり、アナログの代表格としての人への依存性は少なくなります。

一方、ソフトウェア製品の場合、開発完了後は量産化に相当する取組みは求められません。

量産技術の存在が工業製品とソフトウェア製品を明確に区別します。ただし、ハードウェアの中に組み込まれることが多い組込みソフトウェアはグレーゾーンの製品といえます。

2-2.ソフトウェア製品

ソフトウェア開発のV字モデルを図3に示します。

このように図的に表現するとソフトウェアは技術として確立されているかのように見えます。

しかし、ソフトウェアを開発する段階で人依存型の構造が露呈します。ソフトウェア開発あるいは改修においては、担当する技術者がドメイン側の知識や暗黙知、ソフトウェアの開発経験などをもっているか否かで、品質と開発効率は大きく左右されます。一方で、長年に渡って改修を重ねてきた大規模なソフトウェアは改修の度にノウハウを詰め込んできており、全体を新しく作り直すことは大きなリスクを抱え込むことになります。同じ意味で全体に渡る大規模な改修も困難なのです。このようにソフトウェアの人依存性は解消されず、ソフトウェアの中に経験知や暗黙知といったアナログ的特性が内包されているのです。

2-3.ソフトウェアはイノベーションの阻害要因!?

ここで大きな問題が見えてきます。

身動きが取れない過去のプログラムを継承するソフトウェア資産は、新しいシステムに脱皮できなくなるのです。改善改良をしていく持続的なイノベーションも創造的破壊型イノベーションも起こせない「イノベーションのジレンマ」と同じく、いわば「ソフトウェアのジレンマ」ともいえる状態に陥る可能性が高いのです。

日本には通信、製造、金融、医療、交通、流通等々の産業・サービスが存在しています。これらの産業やサービスは互いに好循環の関係によってイノベーションを起こしていくものと誰もが期待しています。しかし、これらの産業のソフトウェアへの依存度は増加の一途を辿っています。

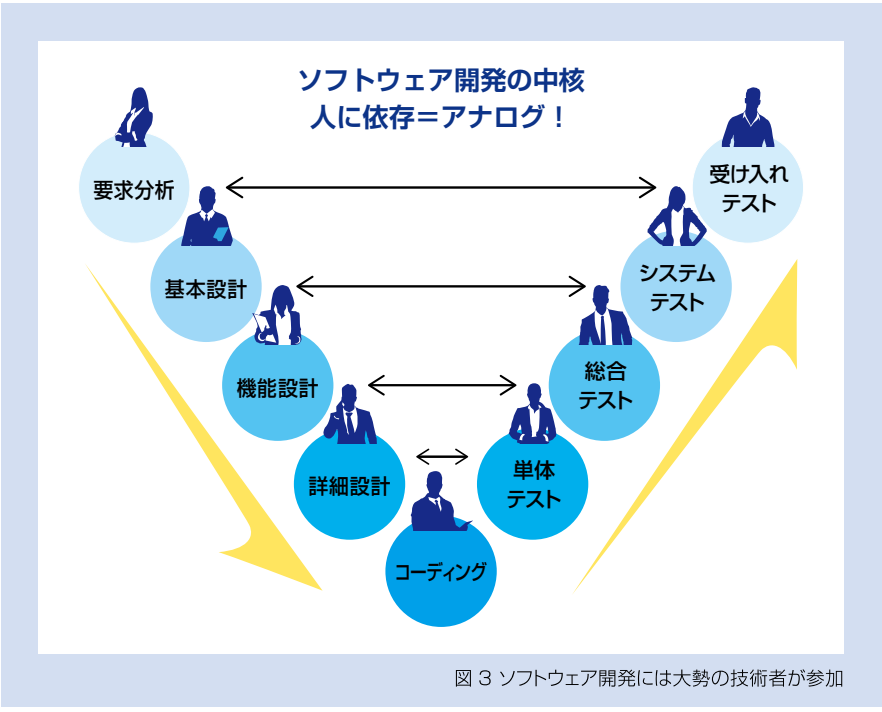


図 3 ソフトウェア開発には大勢の技術者が参加

アナログ的なソフトウェア開発への適切な手を打たない限り、これらの産業はソフトウェアのジレンマに巻き込まれることになります。ソフトウェアのみでなく日本型の経営もアナログのままになっているのではないかと危惧されます。アナログ型文化のままでイノベーションを起こそうとしても、デジタル型の文化を凌駕する創造的破壊型イノベーションは起こせそうにありません。

3. ソフトウェア開発のデジタル化!

アナログ依存型のソフトウェアに対してデジタルイノベーションは起こせるのでしょうか。結論から言えば、自前主義を捨てたオープンソース、AI、IoTなどを的確に導入することでイノベーションを起こせる可能性が高くなります。

例えば、AIによる創造的破壊型イノベーションの概要を図4に示します。

入力データと出力データの対応関係を学習したAIは、例え学習していないデータが入力されても、そのデータに近い学習済の入力データを見つけ出し、対応する出力データを出力します。その際、出力の確からしさ(確信度)を対にして出力する柔軟性をもっています。ただし、AIの学習用入出力データに誤りなどが入り込むと学習精度が落ちるため、入出力データの正しさを確保するためのデータのクレンジングという作業が必要になります。このように入出力データ作成、AIのアーキテクチャの設計、補助的なプログラム作成などの作業は必要ですが、AIの学習に入るとある意味一瞬で終わります。しかも、入出力データに変更が発生した場合あるいは対応関係が変わる場合でも再学習させることで、新たなAIとして機能します。

AIの学習プロセスは手続き(How to)をアウトプットするソフトウェア開発工程の多くの部分をデータのクレンジングを含むデータ(What)の設計に置き換えてしまいます。さらに経験値や暗黙知は学習用のデータの中に入り込んでいるため、形式化の作業が不要となります。学習させたAIのマッピング機能はテスト済のプログラムに相当します。学習用のデータ作成は発注側(ドメイン側)の仕事となります。結果的にソフトウェア開発側の工程は劇的に簡略化されます。すなわち、ソフトウェア開発の人依存性というアナログ特性は大幅に低下し、創造的破壊型のデジタルイノベーションを加速していくことになります。

後編に向けて

後編では、AIの活用を通してデジタルイノベーションを理解するために、株式会社オプテックが日本マイクロソフト株式会社および株式会社クレスコの協力を得て開発した歯科業界初の歯科医療用AIを例にデジタルイノベーションについて解説します。

※参考文献は後編にまとめて掲載いたします。

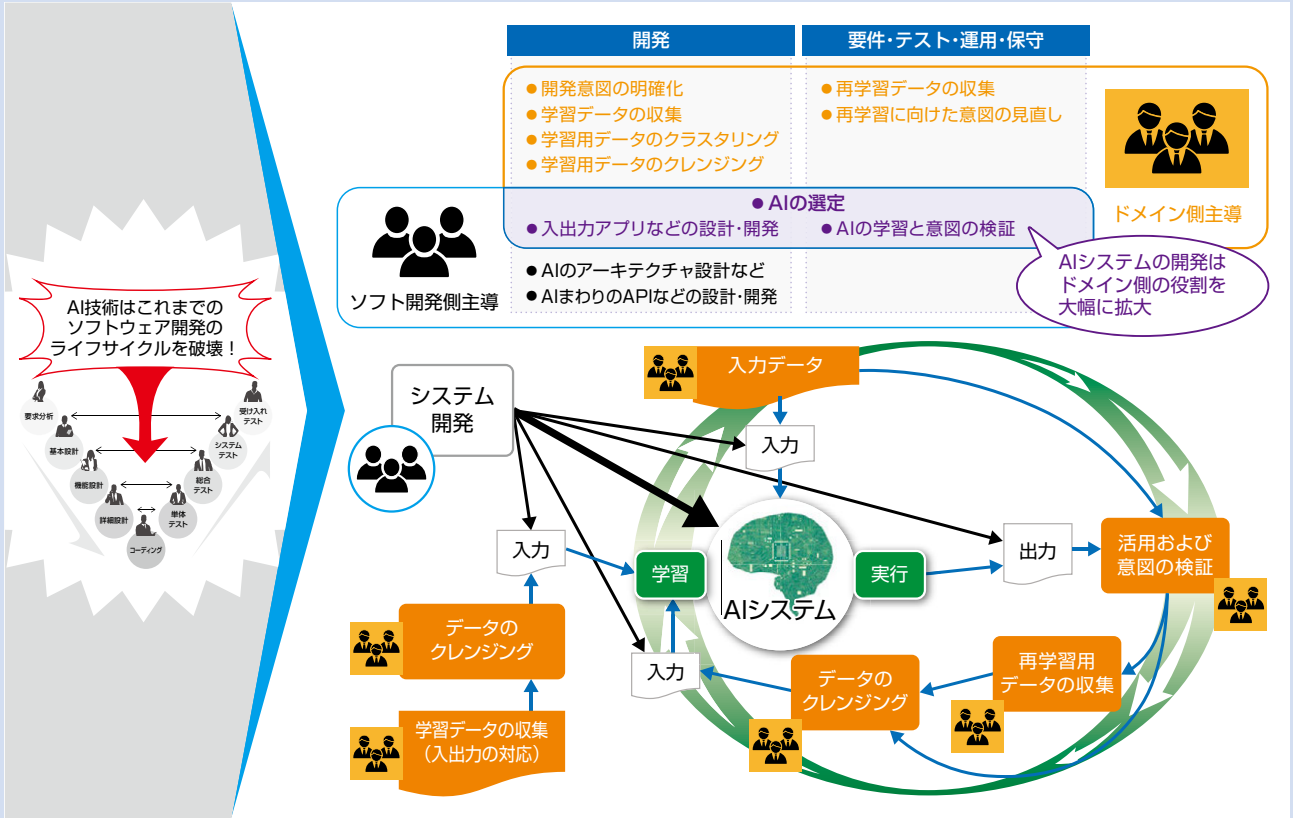


図4 ソフトウェア開発のデジタルイノベーション

本記事の内容に関するご質問、お問合せは、ベリサーブマーケティング部 Email:verinavi@veriserve.co.jp までお寄せください。

お客様のニーズに寄り添うPMOの在り方について

株式会社ベリサーブ IT システム事業部　鈴木 三紀夫

はじめに

当社（株式会社ベリサーブ）はさまざまな検証サービスを提供しており、その中に検証PMOサービス（テストマネジメント支援サービス）というテストを中心としたPMOサービスがあります。しかし、本稿ではプロジェクト全体を対象としたPMO[*1]業務に関する事項についてご説明します。

現状

お客様が手がけるプロジェクトの規模が大きくなり、プロジェクトマネジメントの難易度が上がっています。そのため、プロジェクトマネジャーが一人でプロジェクトを管理するのが難しいと感じるお客様が増えています。その結果、ITシステム事業部でもPMOに関する引き合いが増えています。

過去にも、プロジェクトマネジャーを補佐する役割の人たちはいました。たとえば、次代のプロジェクトマネジャー候補者を補佐役に付けて育成するケースです。これら補佐役をPMOとみなせば、PMOは昔から存在しており新しい概念ではないとみなす識者もいます。

しかし、プロジェクトマネジメントの難易度が高くなるとともに、現在のPMOはプロジェクトマネジメントに特化したスキルを持つ専門家という認識が一般的になりました。このような背景のもと、プロジェクトマネジメントの支援を専門に行うPMOを外部から調達するというスタイルが広まってきました。

PMOとは

PMOの定義を、『プロジェクトマネジメント知識体系ガイド PMBOKガイド 第6版』から引用します。

> プロジェクトマネジメント・オフィス（PMO）は、プロジェクトに関連するガバナンス・プロセスを標準化し、資源、方法論、ツールおよび技法の共有を促進する組織構造である。PMOの責任は、プロジェクトマネジメントの支援機能を提供することから、ひとつ以上のプロジェクトを直接マネジメントすることまで広範囲にわたる。
>
> （中略）
>
> PMOの形態、機能、体制は、支援する組織のニーズによって決まる。
>
> （中略）
>
> PMOの最も重要な機能は、さまざまな方法でプロジェクト・マネージャーを支援することである。

(2018)『プロジェクトマネジメント知識体系ガイド PMBOKガイド 第6版』pp.48-49, Project Management Inst

*1: PMOとはProject Management Officeの略です。複数のプロジェクトを横断的・統合的に管理する場合には、Program Management Office の略としている組織もありますが、区別していない組織もあります。なお、本稿では区別していません。

PMOの主たる仕事は、プロジェクトのマネジメント業務をサポートすることです。スケジュール作成を支援したり、進捗管理や品質管理などのマネジメント業務の一端を担ったりすることから、ある組織ではPMOを後方支援部隊と呼ぶこともあります。

PMBOKガイドでの定義にもあるように、PMOは組織のニーズによって形態、機能、体制が変わります。あるお客様のニーズは、プロジェクトマネジャーの育成が急務であるため、プロジェクトマネジャー候補者とPMOが一緒に働くことで必要なスキルを身に付けさせたいというものでした。また、別のお客様のニーズは、経営層にプロジェクトの状況が正しく伝わっていないのではないか、という経営層の問題意識から、第三者的な視点により現状を正確に報告して欲しいというものでした。

このように、あらゆる組織のニーズを満たし問題を解決するには、PMOのような役割が必要となり、ニーズに合わせて機能や体制を提供するようになりました。

プロジェクトマネジャーとの違い

PMOはプロジェクトマネジャーが持つかなりの業務を支援し、代わりに遂行することができますが、PMOにはできない業務があります。それは、プロジェクトの意思決定に関わる業務、メンバーへの指示・命令を行うリーダー的な業務です。

PMOは必要に応じて、プロジェクト情報の収集、状況の把握、問題の識別、原因の分析、解決策の提案、チーム間の調整などさまざまな業務を行いますが、解決策をどれにするかという意思決定とその解決策を実行するための指示・命令は、プロジェクトマネジャーしかできません。

PMOの目的と業務

PMOを設置する目的はプロジェクトによって異なります。多くの場合は、プロジェクトのQCD目標の達成に寄与することです。プロジェクトを円滑に効率的に運営することを目的にする場合もあります。

目的を達成するために、PMOはさまざまな業務に従事します。主な業務を以下に記します（表1）。

業務名	主な内容
事務的な業務	プロジェクトデータの収集、更新、共有 各種書類作成・申請業務
プロジェクトマネジメント業務	スケジュール作成　進捗管理　品質管理　課題管理 リスク管理　仕様変更管理　構成管理等
標準策定、ツール開発、導入支援業務	プロジェクト管理標準の策定、および文書化 プロジェクトに必要なツールの開発 ツールの使用方法の教育等の導入支援
ステークホルダー間の調整業務	ステークホルダーとの伝達業務 ステークホルダー間の調整業務
プロジェクトマネジャー補佐	プロジェクトマネジャーの参謀役 プロジェクトマネジャーの代行
情報報告業務	プロジェクト状況の把握 意思決定の参考になる情報の提供

（表1）PMOの主たる業務

PMOの形態

PMOは組織の中のどこに設置するかによって形態が変わります。形態とは、プロジェクトとPMOの関わり方を指しています。さまざまなPMOの導入形態がありますが、次の3種類の形態がよく見られます（表2）。

PMOの機能

組織によっては、PMOの業務が明確になっておらず、状況に対して臨機応変に対応することを期待している場合があります。このような状況の場合、PMOの機能（役割）として認識した方が分かりやすいかもしれません。さまざまな機能を次のように整理します（表3）

形態	主な内容
単独のプロジェクトに参画するPMO	特定プロジェクトの体制に入り、事務的、管理業務の全般を行う
複数のプロジェクトの支援をするPMO	定期的な管理作業など、特定の支援を複数プロジェクトに対して行う
組織の全プロジェクトをモニタリングするPMO	組織内の全プロジェクトを監視し、支援が必要なプロジェクトを早期に指摘するなど、横断的な視点での助言を行う

（表2）PMOの形態

サポート機能

プロジェクト運営を支える定常的な情報収集、各種書類作成などの事務的な業務を行う

ミツバチ機能*2

プロジェクト外の知見を持ち込む
知見をプロジェクトに合わせてカスタマイズしたり、メンバーにレクチャーしたりする

ペースメーカー機能

計画策定を支援する
定期的な会議開催を通じて進捗状況、課題・リスク状況を把握し、ボトルネック発見と解消を手伝う

リエゾン*3機能

プロジェクト間、チーム間を仲介する
チーム間の橋渡しを行い、各種事項を調整する

アラーム機能

必ずしも定期報告には表れない、各種プロジェクトデータの分析によって、プロジェクト運営上の阻害要因の発見と警告を行う

コンサルテーション機能

プロジェクトに潜むさまざまな問題に対して解決への道筋を見つけ出す
各種知見や解決策を押しつけるのではなく、プロジェクトマネジャーと一緒になって考えるよき相談者という位置づけである

ブレーキ機能

プロジェクトの状況を踏まえて、プロジェクトを立ち止めることを進言する
たとえば、不具合が多くテストケースの消化が思わしくないとき、前工程まで戻ることを進言するなど

（表3）PMOの機能

*2: ある花の花粉を別の花に運び受粉を助けるのがミツバチである。社内外の優れたプロセスや、手法をプロジェクトに持ち込み、プロジェクトの成長を促すことをミツバチ機能と呼ぶ。
*3: フランス語で「結びつき」を表すことから転じて、連絡、連携などを表す。

PMOの事例

このようにPMOはどのような仕事をしているのかご説明しましたが、イメージは難しいかもしれません。理解を促すために2つほど事例をご紹介します。

事例1

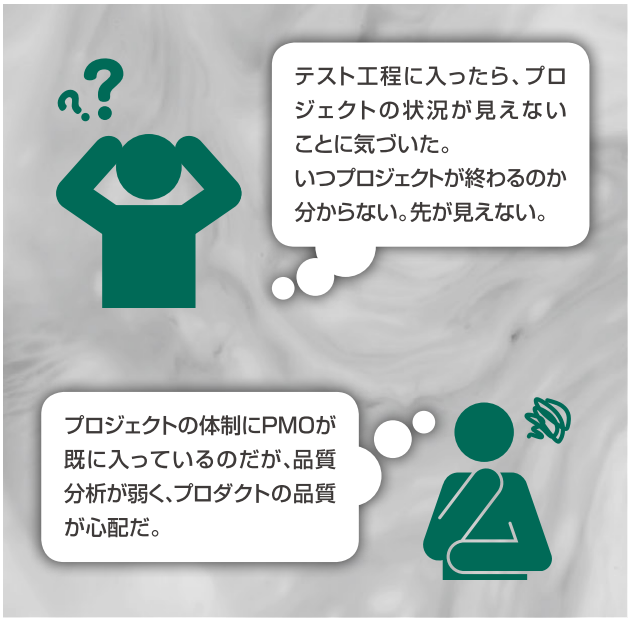
お客様から「不具合が多く、テストの進捗遅れが発生しているが、どこに問題があるのか分からない。」という相談を受けました。

不具合分析を実施し、問題箇所の絞り込みを行ったところ、あるアプリケーション機能のテスト不足が明らかになりました。さらに原因分析を行ったところ、構成管理[*4]の不備が分かりました。このケースでは、追加テストの提案だけでなく、構成管理ツールの導入から構成管理ルールの定着までを支援しました。

事例2

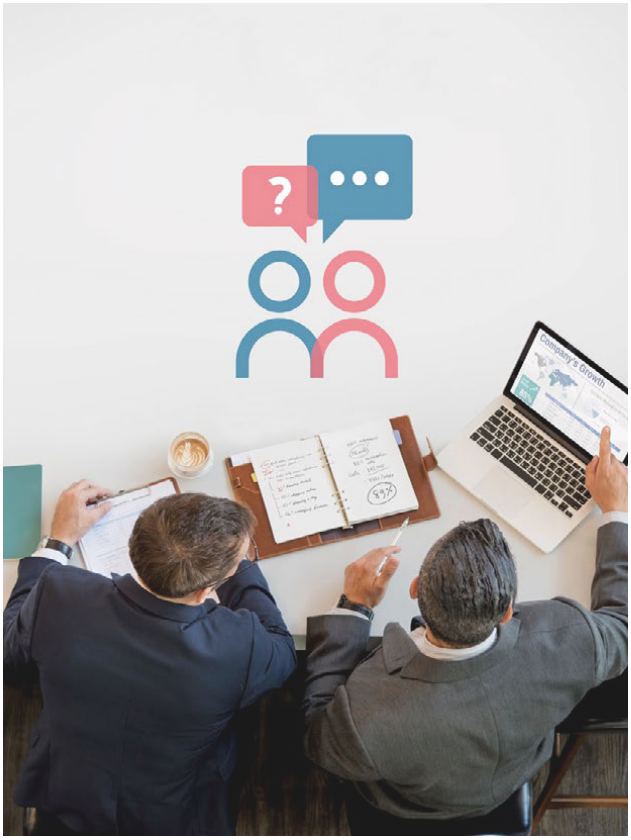
お客様から「各プロジェクトの体制には既にPMOがいるのだが、品質分析が弱く、経営層にプロジェクトの状況が正しく伝わっていないのではないか。」という相談を受けました。

手始めに適度な大きさのプロジェクトの品質報告書を見たところ、不具合の分類は行われていますが、分析は行われていません。そこで、複数のプロジェクトの品質分析を並行して行い、その問題点を見つけ出し、経営層に報告をしました。



おわりに

検証サービスのイメージが強いベリサーブが行っているPMO業務について紹介をしました。今ベリサーブに求められているPMOとは、お客様のニーズに寄り添い、PMOとしてどのように振舞うのが適切であるか意識しながら支援できる存在であると考えております。ベリサーブではこれまでに培った検証サービスのノウハウと、ご紹介しましたPMOの機能を組み合わせて、さまざまな形でお客様の課題解決をご支援いたします。



*4: ソースコードや文書などの成果物の変更履歴を管理すること。

本記事の内容に関するご質問、お問合せは、Email:vs.marketing@veriserve.co.jp までお寄せください。

「ニアショア検証サービス」のご紹介

はじめに

国内のITサービスの市場は、AI、IoT、ビッグデータ、クラウド、ソーシャル、モバイルなどの"第3のプラットフォーム"と呼ばれる分野の需要が急速に伸びています。一方2020年～2030年にかけて20～40万人を超えるITエンジニアが不足するという試算結果が出ており、IT技術の移り変わりが早い市場において、深刻なエンジニア不足が懸念されています。

このように自前で要員を確保することが非常に困難な中、エンジニア不足を補う手段の一つとして、外部の専門会社との連携が不可欠となっており、その連携の形態として、アウトソーシングや外部委託先として「ニアショア」をご選択いただくケースが非常に増えております。

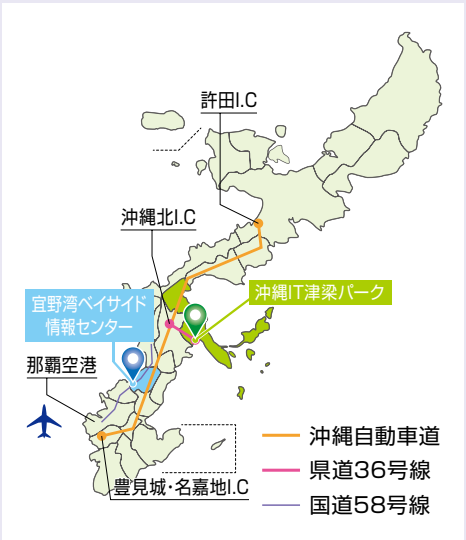
そこで、QCD（品質/コスト/納期）の課題を解決する手段として、「ニアショア検証サービス」のご紹介をさせていただきます。

当社のニアショア拠点について

当社の「ニアショア検証サービス」は、沖縄に拠点を置いている「株式会社ベリサーブ沖縄テストセンター（以下、VOTC）」が行います。VOTCは当社の完全子会社であり、長年のテスト、検証技術を背景に、高いプロジェクト遂行能力を保持しております。また前述の深刻なエンジニア不足への対応として地元の優秀な人材の安定的な雇用を確保しており、大都市圏と比較して割安感のあるコストパフォーマンスを実現しております。そのような環境の中で、お客様は、多数のプロジェクトを同時進行することが可能であり、昨今の天候不順や災害時のダメージを分散化することで、プロジェクトの継続性が確保できるのも大きなメリットです。

一方、オフショアも上記のようなメリットはありますが、言語の制約、開発環境の相違、現地責任者の選定や育成に時間がかかること、カントリーリスクや為替変動リスクなど、ある程度の経験と状況の見極めが必要になってきます。

（図 1）株式会社ベリサーブ沖縄テストセンターの拠点



<うるま本社（沖縄IT津梁パーク内）>

<宜野湾サイト>

沖縄 IT 津梁パーク

- 約700平米（全体1200平米）
- PRJルーム・MTGルームを計14構築
- シールドルーム完備

宜野湾ベイサイド情報センター

- 約230平米
- プロジェクトルーム　3部屋
- ミーティングルーム　1部屋
- 研修施設（有料）

※2018年11月1日 時点

沖縄県は、情報通信関連産業の振興や推進を目的に、さまざまな支援制度が充実しています。それらを活用する形で、沖縄県内のIT産業施設である2拠点にてサービスを提供しております。(図1)

当社ニアショア検証の特徴と活用するメリット

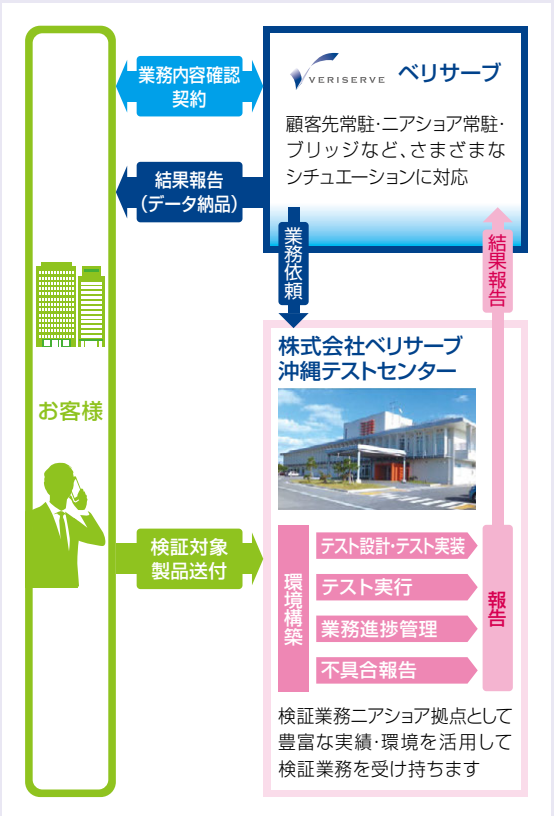
(1)検証専門会社のQualityを維持した、高いプロジェクト遂行力

当社が長年蓄積してきた検証技術や検証実績をVOTCへ水平展開し、当社の検証ノウハウを最大限活用することで、プロジェクト計画・管理水準の共通化やさまざまなドメイン対応実績の共有化などを実施しています。

例えば、自動車分野のECUに関する知識や"CANoe"などのツール利用技術など、ビジネスドメイン別に特化した知識や技術を有する技術者の集中育成を実施しています。

ニアショア活用のプロセスとしては、当社がお客様と「業務契約・要件確認・結果報告(データ納品)」を直接行い、当社とVOTCの業務連携による共同プロジェクトチームを推進体制とし、検証技術支援やプロジェクト/アウトプットの管理を行うことで、お客様のニーズに柔軟かつ迅速な対応が可能です。(図2)

(図 2) ニアショア活用プロセス



(2)安定したCost PerformanceとDelivery効率向上の実現

沖縄県が用意するさまざまな支援メニューを活用及び連携することで、大都市圏と比較して競争力のある価格帯で検証サービスのご提供が可能です。

- 高セキュリティ支援設備を拠点とした、テスト環境の構築
- 豊富に取り揃えた機材の利用と集約拠点への効率的な調達
  国内発売携帯端末を網羅設置、Bluetooth・Wi-Fiプロトコルアナライザ導入、情報家電・自動車・車載系の評価用機材多数保有など
- 業務の山谷に対応した柔軟なリソース調整
- 県の支援を活用した人材雇用と人材育成

ご利用実績の紹介

人材調達、高い検証品質や環境構築などのメリットにより、これまでさまざまな業界やドメイン、規模、期間のご利用実績があり、ニアショア検証拠点としてのご利用が増加しております。また自動車分野、デジタル機器分野のシステムテストを中心に機器間や多数の携帯端末との接続、クロスブラウザやOSなどの互換性検証など、各種の検証業務を継続的にご依頼いただいております。

検証実績(機能検証 設計／実行)

- ナビ
- 自動車ECU
- ネイティブアプリ
- Webアプリ ECサイト
- デジタル機器
- メディカル

今後の展開

IT人材の需要がますます高まる近年の市場変化に対応し、より総合的な検証業務の提供が必要な中、ニアショア検証拠点として期待される「人材調達力」や「検証能力」の確保、継続的なサービス提供を実現させる為の「検証技術の高度化」、それらを下支えする「人材採用」「人材教育・研修」を行い、ニアショアならではのサービスや技術の拡充を図ってまいります。一方で、地域の雇用創出・ビジネス人材の高度化を先導し展開することで「人が集まる地方のIT業界づくり」を実現し、持続可能な地域社会の発展に貢献してまいります。

お問い合わせ先
拠点の視察も随時受け付けております。「ニアショア検証サービス」にご興味がございましたら、以下までご連絡をお願いいたします。
株式会社ベリサーブ ニアショア推進部 担当窓口(佐藤了一)
TEL：03-5909-5700
E-mail：ryouichi.satou@veriserve.co.jp

VERISERVE NAVIGATION（ベリサーブナビゲーション）

2018年11月号

編集・発行	株式会社ベリサーブ 東京都新宿区西新宿6-24-1 西新宿三井ビル14F
お問い合わせ	マーケティング部：03-6302-0707 verinavi@veriserve.co.jp