8.5 品質保証（QA）部門の役割は、・・・微妙

アジャイル開発を製造請負契約で受注しないことを監視する必要はあります。 しかし、製造請負契約のウォーターフォール開発に比べ、準委任契約で行うアジャイル開発での QA の出番は明確ではなさそうです。 もし、存在感を発揮するとすれば、以下の対応が必要でしょう。

■ PO の視点でアジャイル開発を観測し、問題を抽出しましょう。

そのためには毎朝のミーティングに参加して、チーム状態を観測する必要もあります。

■ 発注側の PO にアドバイスしましょう。

アジャイル開発の進め方の問題があれば、PO に対して意見する必要があります。
スクラム開発には、スクラム・マスターというポジションがありますが、それに近いポジションにならざるを得ないかもしれません。
しかし、ウォーターフォール開発の時のような活躍の場があるのかについては、不明です。

WF 発注側 | AG 発注側
WF 受注側 | AG 受注側

9. 和アジャイルの誰時

9.1 本書をざっと、まとめ直してみる

再び本書の目次を眺めてみてください。 発注側の経営幹部、PO、受注側の開発メンバ、受注側の責任者に向けて、それぞれに対するメッセージとしています。 表紙にも開発ステップに関するメッセージを埋め込んでいます。

■ 発注側・経営幹部へのメッセージ

- **アジャイル開発のバリットを受けるのはあなた（経営幹部）です。** ソフトウェア開発の進め方を、**「大きなことを先にやる」**から**「フィードバックを早く回す」**、そう変えるだけです。 変化する状況にスピーディに対応するには、その変化対応力が大切なのです。
- アジャイル開発を推進するのはあなた自身（経営幹部）の役割です。 あなたが自らの組織を変革していかなければいけません。もし、社内でアジャイル開発が未だ上手く実践されていないとしたら、組織風土（価値観、責任・制度、育成）に問題があるはずです。 それを**改革するために、先頭に立って汗をかいてリーダシップを見せていかなければいけないのは、あなた自身です。 あなたはまず PO の心得を知らなければいけないでしょう。**
- 開発責任者になれるPOを育成し、支援し、援護しなければいけません。 PO とは情報処理技術を知り、チーム運営のできるビジネス・リーダです。

■ 発注側・PO へのメッセージ

- **アジャイル開発は準委任契約で実施**しないと、良いソフトウェア作成には至りません。 請負開発契約でアジャイル開発を実施した場合には、あなたの期待するアジャイル開発になることはあり得ません。
- アジャイル開発では **PO に完成責任・品質確保責任**が発生します。 責任に見合う権限を持ちましょう。
- PO に必要なのは**ビジネス・リーダシップによるチーム運営力**です。 PO が受身だとソフトウェアは完成しません。 主体性が重要です。
- 良いソフトウェアを作るには開発メンバのモチベーションが重要です。 **モチベーションを高く維持するために PO は努力**しなければいけません。

■ アジャイル開発メンバ（ソフトウェア技術者）へのメッセージ

- 「アジャイル開発をやりたい」と言うのは発注側です。 受注側が言い出し誘導するのは無責任です。 なぜなら、発注者側がソフトウェアの完成責任・品質確保責任を負うべきだからです。
- アジャイル開発に参加したいが契約制度に関わりたくないなら、自社製品の内製プロジェクトや、派遣社員として他社のアジャイル開発プロジェクトに参加するのが、簡単な道です。
- アジャイル開発では、**コミュニケーション・スキルが重要**になります。 コミュニケーションをどうしても避けたければ、アジャイル開発からは離脱しましょう。
- アジャイル開発では、PO の優先度判断によってプロジェクトが進みます。 **目指すゴールと PO の価値観を正しく理解**して、あなたの**主体性を発揮**して PO を支援していきましょう。
- アジャイル開発に参加したからには、技術的な貢献をしましょう。 プロジェクト以外で**情報処理分野の最新技術を広く習得し、自分を磨き続ける姿勢**を持続しましょう。

■ 受注側責任者へのメッセージ

- アジャイル開発を成功に導くためには、**製造請負契約で受注してはいけません。** 目先の売り上げに負けて製造請負契約で受注すると、両者が、そして日本の情報処理業界全体が、子供たちが不幸になります。

9.2 和アジャイルが目指すべき世界

日本におけるアジャイル開発、和アジャイルで目指す世界はどこでしょうか？

- 日本のソフトウェア開発が世界で競争力を持つこと。 情報システムの開発能力が米国並みに上がること。 発注者が望む良いソフトウェアをリーズナブル（価格、期間、品質）に作りだす世界になること。
- モチベーションの高い優秀なソフトウェア技術者が活躍して良いソフトウェアを作れる場が普及すること。 ソフトウェア技術者が開発で充実感と成長を感じられる世界になること。

著者は、これらを目指します。 そのためにはウォーターフォール開発ではダメです。 アジャイル開発を日本における受発注型ソフトウェア開発の現場でも普通に使える状況が必要です。 2020 年度からはプログラミング教育が小学校に導入されます。 彼ら・彼女らが就職する前に日本の現状を変えなければいけません。

■ 日本の現状の不都合を正しく認識する発注者が主流になることが望まれます。

まずは、日米格差とその問題点を発注者が正しく理解することが大切だと考えます。

日本におけるアジャイル開発は、IT 系ベンチャー企業などでは既に常識として取り入れられていると思います。 しかし、メインフレームの経験等のある企業や、発注でシステムを開発しようとする企業では未だ主流の開発スタイルには至っていないのではないでしょうか。 まだまだ、日の出前の状態にあります。 本書名に使った**「誰時」**とは、**明け方・日の出前**のことです。 既にアジャイル開発を実践できているベンチャー企業の皆さん、内製開発をベースにアジャイル開発を実践している皆さんには、本書の内容は違い国の笑い話かもしれません。

■ アジャイル開発スタイルを日本の受発注による ソフトウェア開発の実情に合わせて、上手く実施するための工夫とコツを見つけ出すことです。

漢字は中国で作られました。しかし、日本独自のひらがなを生み出し、漢字の間にそれを挟んで多彩な表現と文化を生み出しました。 米国企業が活用するアジャイル開発を日本の受発注関係の中で使っていくためにも、いくつかの工夫とコツの追加が必要です。 本書ではそれをまとめたいと思い書き始めました。

9.3 本書標も β 版

そして、この書籍自体もアジャイル開発のような継続的な進化を目指します。

■ フィードバックを得て、改善をまわすのが「アジャイル」なスタイルのハズです。

本書も、「アジャイル」なスタイルで出版します。 他の皆さんの意見を聞きながら改善するのがアジャイルスタイルだと信じるからです。 まだまだ日本のソフトウェア産業界におけるアジャイル開発スタイルは「誰時」なのです。 時代の変化・人々の経験からのフィードバックに対応して、進化させ出版していきたいと思います。

・フィードバック先: （省略）
・情報の更新案内: （省略）

■ 著者の限界

著者は、内製経験、アジャイル開発の発注側 PO 経験・受注側経験、ウォーターフォール開発の発注側経験・受注側経験があります。 しかし、その開発規模は1回あたり 3 億を超えたものはありません。 アジャイル開発は大規模になると課題が大きくなると言われています。 本書は、その程度の規模までの経験者が執筆した書籍だと理解してください。 本書はアジャイル開発スタイルとしてはスクラムとスパイラル開発を想定しています。
本書の記載と各主張は、著者の所属する組織とは一切関係がありません。

補足 1 アジャイルソフトウェア開発宣言

この宣言は非常に良く、本質を突いていると思います。

Kent Beck et. al, **Manifesto for Agile Software Development, 2001.**
http://agilemanifesto.org/iso/ja/manifesto.html（2019.03 閲覧）

補足 2 ウォーターフォール開発の見積りとは何だったか？

■ 科学的見積りの存在を否定している訳ではありません。 しかし、低コストで期間に精度良く見積る方法は存在しません。

■ 1週間以内に低スキル・低コストで見積るには、どうするでしょう？

科学的かつ実用的な見積りが無いとき、どうやって見積るのでしょうか？ たとえば、以下のような見積り手法を採っているかもしれません。

案 1: WBS 等を作成して KKD 法（勘、経験、度胸）で見積る。
案 1-1: 案 1 ＋ 複数の人の KKD 法を合わせて議論し妥協点を見つける。
案 1-2: 案 1 ＋ 前提条件を非常に細かく多数付けて、契約変更に誘導する。
案 2: 同一条件で下請け社に KKD 法で見積ってもらい、それに自社利益を加えたものを見積りとする。 受注後はその下請け社に製造請負開発で外注する。【ゼネコンモデル】
案 3: 優秀な営業が、競合社や顧客財布事情を見極めて見積る。 到達点は受注後になんとか調整する。

しかし、正確な見積りができないことを認め、見積りをいくつかのフェーズに分けて、都度見積りを修正する見積（再見積・契約変更）を段けるのが、受注側にとって最も合理的な進め方と考えられています。

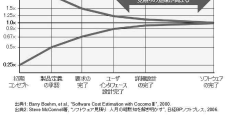
■ 建築業界を素朴に真似たプロジェクト管理と見積りは無茶です。

ゼネコンモデルに従って、建築業界のプロジェクト管理を素朴に真似て進めることも多いでしょう。 しかし、差分をきちんと理解し現実を見直してみましょう。 物理制約の全く無い情報処理の世界、日々新たな技術が登場する世界、そんなソフトウェア製造の世界を、建築とのアナロジーだけで進めることには無理があります。

■ 「不確実性コーン」について理解しましょう。

ソフトウェア開発の見積りについて、科学的な調査の一つである「不確実性コーン」を知ることは大切です。

不確実性コーン（ソフトウェア見積りの誤差）
※ 見積りの精度が最良の正確な見積りをした場合の見積り誤差



出典1: Barry Boehm, et al. "Software Cost Estimation with Cocomo II", 2000.
出典2: Steve McConnell著「コードコンプリート第2版」上下巻(翻訳対応版)ギヴ・ジャパン/ソフトプレス, 2004.

不確実性コーンは、ソフトウェア開発の見積りは開発フェーズが進むと不確実性が減って見積り精度が**急激**に良くなっていくことを示します。 逆に言えば**初期の見積り誤差は十数倍の見積り誤差を含むのが普通だ**ということも示します。 それも、見積りをするのは**「熟練した見積り者」が「最良の正確な見積り」を実施するという前提**です。 別の見方をすれば、たとえば以下のようなことが言えるでしょう。

- 初期段階の見積りは、正しい見積り（実際にかかる費用）の 10 分の1である可能性が**普通に**ある。 その逆にある。
- 初期段階で、1週間かけた見積り精度と、2 週間かけた見積りとの精度での見積り精度向上に大きな差は出ない（つまり、見積りの金額差は出るかもしれないが、本質的に同程度の誤差が含まれる）。 それよりも・・・
- 作成するソフトウェアの不確実性を下げることによって、見積りの精度が高まっていく。 **全てのユーザインタフェースの設計が完了**した時点で、±25%程度の誤差以内に収まることが期待される。 注意点を繰り返すと、これは「熟練した見積り者」が見積る場合である。
- 詳細設計の完了もしくは全ユーザインタフェースの設計が完了からソフトウェア製造の完了までの期間が十分長い（その期間のコストが支配的）なプロジェクトであれば、熟練者による精度の高い見積りをきちんと行って開発が進められる可能性がある。

この科学的な現実をあなたは素直に見つめることができるでしょうか？ この科学的結果を理解したあなたは、全てのユーザインタフェースの設計が決まる前の見積り・非熟達者による見積りの精度について、どんなことが言えるでしょうか？
詳細は、次の書籍に譲ります。 見積りを語るためには一読をおすすめします。

Steve McConnell 著,「ソフトウェア見積り 人月の暗黙知を解き明かす」, 日経 BP ソフトプレス, 2006.

金額固定・納期固定が前提の契約で見積りも精度を高められないなら、受注側はリスクをそれなりに積むでしょう。 発注側から見てそのリスクが高すぎるか否かも科学的に見れば不毛な議論です。 かといって、早期に自らの不確実性を削減するにも限界があるでしょう。 更に加えれば、市場競争に必要なソフトウェア開発なら、市場の声も大切ですし、ライバルに勝ち一への即応も必要です。

それなら・・・、

■ 稼働実績ベースに支払いをするアジャイル開発でいいんじゃないですか？

アジャイル開発の方が費用根拠は合理的なんじゃないですか？ 最も優秀な開発メンバを研鑽っても精度が高まらない見積りに使うというのは、正しい選択なのでしょうか？

補足 3 ウォーターフォール開発の品質とは何だったか？

ウォーターフォール開発では、**品質を作りこな**という考え方がありました。 それは、工程毎に良い品質のものを積み上げていくことによって実現されるものでした。 しかし、以下のような問いにどう答えるのでしょうか？

- 試験目標密度の目標値には、そもそもどんな意味があったのか？（その意味は今回の開発で正しい基準だいえる根拠があるのか？それは何か？）
- バグ収束曲線は本当に科学的に作られているか？ 収束が志意的ではないか？ 予め抽出した試験項目に対応するバグだけを本当にカウントしているか？ 試験外や試験ついでに見つけたバグを一緒に検出数に含めていないか？ 試験項目消化速度は一定か？ 実は両面赤など、短時間でできるものを最初に実施し、異常系など試験準備が必要な項目を後に実施していたりしないか？ 横軸のメモリ間隔の均一性が担保されているか？ 試験はランダムに配置しているのか？（← プロジェクト効率化上は止めた方がいいのだが・・・）
- バグの原因分析、なぜなぜ分析は、真の原因に辿り着いていたのか？ 「考慮モレ」のレベルで止まっていたりしないか？ 水平展開範囲は真に適切な範囲を選択し、本質的な対処をしていたのか？
- C0/C1 などの考え方は機械的で客観的だが、本当に 100%が最重要なのか？（それよりも優先すべきことが本当に無かったのか？）
- ソースコードのクロスレビューは実施されていたのか？
- コーディング者は、品質確保のモチベーションを持って作成していたか？ 開発者のモチベーションが低下した状態で、品質指標目標だけが一人歩きしていたりしないか？

そして、

- ウォーターフォール開発をきちんと運営することで、重大故障は防げたのか？
- やらないよりはやった方が少し安心だとして、費用対効果は満足できるものだったのか？

唯一の正解はありません。 ウォーターフォール開発の信者であっても、答えにくいものが含まれているのではないでしょうか？

補足 4 準委任契約と派遣契約の違いに注意

準委任契約と派遣契約には、大きな違いがあります。 準委任契約でアジャイル開発を実施する場合には、PO が受注側管理責任者でない・開発メンバに直接指揮命令をすることができないという制約があります。 PO は必ず受注側の管理責任者に対して指示を行うように注意しなければいけません。 なお派遣契約にもここに記載されていない大きな制約がありますのでご注意ください。

	準委任契約	派遣契約
指揮命令	発注側責任者が受注側責任者に依頼、受注側の管理責任者が担当者に指示。（PO が開発メンバに直接指示は できない。）	派遣先の指揮命令者（契約対象窓口）。派遣先責任者も別途指定する必要あり（管理相談先）。
法律	業務委託契約	労働者派遣契約
義務	完成責任なし。善管注意保義任なし。善良な管理者の注意義務あり（専門家としての能力、注意義務）	完成責任なし。善管注意保義任なし
勤怠管理（起動、休暇）	受注会社の勤務管理者	派遣先の指揮命令者

※ 著者の知識が不正確な場合があります。正確には各法律を確認してください。
※ 準委任契約で指揮命令系の違反があると偽装請負になりますので特に注意が必要です。