和アジャイルの誰時 β2

日本で実施する発注型アジャイル開発

Step1 まず発注側がアジャイル無識に変革すべし。

アジャイル開発のメリットを享受するのは発注側の経営者である。
ソフトウェア開発の現実を知り、日本の格差を知って、
発注側経営者は組織価値観を変革し、制度を適合させ、
PO 人材を育成し、援護すべし。

Step2 発注者は開発者から信頼された仲間になるべし。

責任押し付け、予定実績管理するだけでは良い開発はできない。
ゴールを示し、皆に理解してもらってチームを作り、
一人ひとりを知って適材適所に配し、
状況・問題点・今後の方針を積極的に聞き、対外調整を実施し、
丁寧な対話とバランスの良い優先度判断を常に心がけ、
仲間であり人格者のリーダとして信頼されるようになり、
仲間ともに持続的な Win-Win と成長を目指すべし。

Step3 受注者全員が日々、全体と優先度を考え行動すべし。

真のゴールと PO の価値観を理解し、チームの現状を把握し、
自らの状況を適切な相手に適切なタイミングで伝え、
次の作業案をもって優先度を都度 PO と相談し、
技術者としての高い付加価値を提供し続けるべし。

目次

1. 日本におけるアジャイル開発 1
1.1 アジャイル開発を、誰が、なぜ、求めるのか？ 1
1.2 皆が認めるべき、ソフトウェア開発の現実 3
1.3 ウォーターフォール開発では、なぜダメなのか？ 7
1.4 日本の慣習で、なぜアジャイル開発は失敗するのか？ 9

2. 発注側 経営幹部の心得 17
2.1 アジャイル開発は、あなたの望みを実現する手段 17
2.2 組織をアジャイル開発に適合させる 20
2.3 PO 人材を育成し、援護する 23
2.4 アジャイル開発の不都合を理解する 29

3. 発注側 PO の心得 33
3.1 プロダクトオーナ(PO)の責任と権限を理解し納得する 33
3.2 発注側経営部がアジャイル開発の価値観に納得していない時 38
3.3 ゴールに対して、価値のある部分をまず動かして全体の目処をつける 41
3.4 動いたら、ステークホルダに見せ、アジャイル開発の価値を活かす 44
3.5 その後も「動くものを作る」を忘れずに、改善のサイクルを回す 47
3.6 期待されるのは普通のビジネスソーダジップ 50
3.7 ウォーターフォール思考・姿勢の注意点 51
3.8 コミュニケーションは大切、・・・相談を中心にする 54

4. コミュニケーションの本質を考え直す 55
4.1 コミュニケーションには、双方の努力が必要 55
4.2 相手を知る＝相手の人物モデルを自分の中に持つ 59
4.3 重要なことを先に、段階的に詳細化し相互理解する 62
4.4 モチベーションを上げるために（PO 視点） 63
4.5 チームの質も上げよう（PO 視点） 65
4.6 悪い情報を素早く収集する仕組みを作って、
大失敗を回避する（PO 視点） 66
4.7 ウォーターフォール型コミュニケーションの悪癖 69

5. 現場でのアジャイル開発スタイル
（PO 実践編） 71
5.1 技術者を集める 71
5.2 自らがゴールを示し、皆に理解してもらってチームを作る 76
5.3 一人ひとりを知って適材適所に配置する 78
5.4 状況・問題点・今後の方針を積極的に聞き、現場を正しく理解する 79
5.5 ステークホルダとの調整を実施する 81
5.6 丁寧な対話を続け、バランスの良い優先度判断を続ける 83
5.7 人格者のリーダとして信頼される 84
5.8 仲間とともに Win-Win と成長を目指す 85

6. 品質について考え直す 87
6.1 ウォーターフォール開発では形式的な合意形成の事例がある 87
6.2 アジャイル開発での品質確保・試験終了の考え方 92
6.3【方針 1】 重要な部分から、品質を確保する 94
6.4【方針 2】 継続的に品質を改善する(自動リグレッション・テストの整備) 98
6.5 それでも、納期が厳しく品質も求められる場合の対応(例) 102

7. 開発担当者の心得 105
7.1 真のゴール、PO の価値観、チームの状況を理解し、
PO の判断に近づく努力をする 105
7.2 自らの状況を適切に伝えるコミュニケーション姿勢を持つ 108
7.3 次になすべき仕事案をもって優先度を都度 PO と相談する 111
7.4 技術者として高い付加価値を提供するべし 113
7.5 相互にフィードバックし、双方が成長する 115
7.6 アジャイルの罠に注意する 116
7.7 ウォーターフォールの悪癖は矯正する 118

8. 受注側 開発責任者の心得 123
8.1 アジャイル開発を受注するときに確認する 123
8.2 PO をサポートする 132
8.3 作業管理責任者として行動する 133
8.4 受注側組織の体制・価値観を VerUp する 136
8.5 品質保証(QA)部門の役割は微妙です 139

9. 和アジャイルの誰時 141
9.1 本書をざっと、まとめ直してみる 141
9.2 和アジャイルが目指すべき世界 144
9.3 本書着も β 版 146

補足 1 アジャイルソフトウェア開発宣言 147
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？ 148
補足 3 ウォーターフォール開発の品質とは何だったか？ 152
補足 4 準委任契約と派遣契約の違いに注意 154
補足 5 アジャイル開発におけるリーダシップとは？ 155
補足 6 単能工と多能工、ライン生産とセル生産 156
補足 7 本書で使う工程名とＶ字モデル 157
補足 8 本書の色使い 158

1. 日本におけるアジャイル開発

1.1 アジャイル開発を、誰が、なぜ、求めるのか？

■ アジャイル開発は発注側の経営幹部が「まとめてソフトウェア開発の競争力を実現します。

ソフトウェア開発発注側の経営幹部は、**情報システム(ソフトウェア開発力)を企業競争力の源泉と思っているでしょうか？** そうであれば、その競争力実現のためアジャイル開発の採用を決定するのは間違いないでしょう。 それによって得られるものは、米国企業や国内ベンチャー系企業に対抗できる競争力(開発スピードと変化対応力) です。 この競争力をウォーターフォール開発で得ることは困難です。

■ アジャイル開発の開発スピードや変化対応力よりも、開発リスク低減を重視する非競争組織の経営幹部もいます。

一方、ソフトウェア開発発注側の経営幹部に**市場競争力の向上という価値観**が無い場合や、幹部が発注側の開発リスクを低減することの方が重要だと考える場合には、アジャイル開発の提供する価値は無意味です。 これは例えば、公的機関や共通事務部門の一部に見られるかもしれません。

■ 「優先度の高いところから開発し、動くものを作ってユーザの反応を得て改善を続けていく」という開発スタイルは、経営幹部にとっては普通のビジネス・スタイルと一緒です。

【アジャイル開発の核心】
1. 優先度の高いところから対応し開発していく。
2. 動くものを作ってユーザの反応を得て改善を続ける。
3. 改修したものも修復、動作する状態を維持する。

■ アジャイル開発では、ソフトウェアは「金で買う」から、「チームで一緒に作って育てる」ものになります。

インターネット上でグローバルに競争する時代の経営幹部は、当初の計画のまま1年間進めることなど、考えもしないでしょう。 目指すべきゴールに対して、計画はもちつつも、常にライバルや市場の状況を見て、都度、計画を見直し重要度の高いものから手を打ち、市場の反応を見てまた素早く軌道修正をするという進め方をしているはずです。 アジャイル開発とは、極論すれば、そのビジネス・スタイルをソフトウェア開発でも実践することです。 そのとき、ソフトウェア開発は、**単なる「金で買う」にとから、「チームで一緒に作って育てる」ことに変革している**に違いありません。 駄目ソフトウェアを、時間をかけて安く買っても無駄なのです。 素早い変化対応力が大切です。 そこにこそ高い価値があるのです。 市場の反応を理解し、ソフトウェア開発の優先度を見直し開発をリードする発注側のキーマンを、**プロダクト・オーナ(PO)**とアジャイル開発では呼びます。

1.2 皆が認めるべき、ソフトウェア開発の現実

ソフトウェア開発は工学的に未熟な状態にあります。 その中で良いソフトウェアを作るために理解、認めなければならない幾つかの現実があります。

■ ソフトウェアを混乱少なく開発するには、作りたいソフトウェア（ゴール）の明確化が必要です。

【現実 1】 自分が出したいソフトウェアの具体的なゴール・イメージ（要求仕様）が決まらなければ、どんな開発手法を使っても期待するソフトウェアは作れません。

発注側は、まずはゴールイメージをできるだけ具体化すべきです。 具体化しなければ開発者に伝えることもできません。 ゴールイメージを作るまでは開発を開始しないことが懸命です。 その場合には開発の前に、検討する期間を設けましょう。

アジャイル開発ではゴールイメージが決まり開発を開始した後に、外的要因によってゴールイメージが変化することについて、積極的に許容します。 しかし、それはゴールイメージが無いまま開発を進めることとは異なります。

【現実 1-2】 ゴールイメージ仕様の途中変更を行えば、開発に必要な期間は延びていきます。

ウォーターフォール開発における仕様変更よりはアジャイル開発の方が短期間で対応できることでしょう。 ただし、納期が固定で延伸できない場合には、追加した開発作業量以上の作業量をどこかで削らなければ(ソフトウェアは納期までに完成しないでしょう。

■ 良いソフトウェアは高いスキルと高いモチベーションを持つ人によって開発されます。

【現実 2】 良いソフトウェアは、高いスキルと高いモチベーションを持つ人によって開発されます。 『低良ソフトウェア開発原価(GSDP)』

これを低良ソフトウェア開発原価(GSDP)と呼ぶことにします。 これは十分条件ではなく、必要条件です。 開発者に高いスキルと高いモチベーションが無いと良いソフトウェアは生まれません。 インターネットの普及などと共に情報処理技術はコモディティ化しています。 日本においても教育制度の改革の中で高い情報処理スキルを持つ人は徐々に増えていきます。 内製も困難ではない時代になってきているでしょう。 後は、開発者に高いモチベーションを持ってもらうことが大切なポイントとなります。

【現実 2-2】 アジャイル開発の方が開発者の高いモチベーションを維持しやすく、発注側の期待と実績にあった良いソフトウェアが作成できます。

アジャイル開発の場合には、ゴールに共感する開発メンバによって開発を進めることも可能です。 その時、アジャイル開発はウォーターフォール開発に比べて圧倒的に有利な開発環境を得たことになるでしょう。 ただし、そのためには発注側の負担がより多くなる点には覚悟が必要です。

■ ソフトウェアの科学的かつ実用的見積りは困難です。

【現実 3】 科学的で、実用的なソフトウェア見積り方法は存在しません。

ここで、科学的とは、誰が実施しても期待の見積り精度があるもの、実用的とは、低コスト（数日）で見積り可能なものとします。 科学的と実用的の両方を同時に満たす方法は現時点では存在しません。 この事実を認めたくない人々も今時点の日本には多数存在すると思います。 実際、「これまで、見積りできているではないか？」、「私はパワポ1枚の資料を出しても3日間で見積りをもらって、それで契約して、ちゃんと開発を成功させた！」と、反論する人もいるでしょう。 しかし日本の多くの見積りでは KKD 法を使っていると予想します。 KKD は英単語の略称ではありません。K（勘）、K（経験）、D（度胸）という和製の日本国内限定用語です。

ソフトウェアの見積りに精度については、インターネットなど存在せず技術の進歩がそれほど早くなかった 1980 時代初頭から大きな誤差がでることは示されていました。 これは**不確実性コーン**とも呼ばれています。 詳細は、巻末の補足 2 に譲ります。

KKD 法の中で、唯一使りになるのが、経験の蓄積の活用です。 これまで多くの試みがありました。 しかし、インターネットによって技術の進歩が一段と早くなった現代、さらに多様なソフトウェア形態が求められるようになった現代において、過去データの蓄積を使った期権による見積りは益々困難になっています。 仕様変更に伴う再見積り時であっても一緒です。

不確実性を減らす努力を発注者も協力して行い、時間をかければ見積りの精度も高められます。 しかし、多くの場合にそれはアジャイル開発そのものよりも、だいぶ非効率なものになるでしょう。 まして、仕様変更が多発するソフトウェア開発で精度の低い見積りはただの無駄です。 合理的でロジカルな国の人々は、KKD 法などという意味不明な手法を認めることはなく、アジャイル開発を普及させることになりました。

1.3 ウォーターフォール開発では、なぜダメなのか？

【WF 弱点 1】 アジャイル開発に比べ変化対応力が弱いから、完成した時点でさいものになっています。 競争相手がアジャイル開発の場合には常に遅れ続けることになります。

ウォーターフォール開発(WF)は工程を区切り、前の工程までをきちんと仕上げてから次の工程に進む開発スタイルです。 各工程の成果物をしっかりレビュー し品質を確保します。 V 字モデルに従って成果物とテストとを対応（例: 詳細設計書に単体試験を、基本設計書にシステム結合テストを、要求仕様書にシステム総合テスト対応）をさせて品質の積み上げと呼ばれる作業を行います。

一旦、要件 FIX や仕様 FIX を行うと、その後の工程でそれらを変更することは原則として行いません。 逆流しないことからウォーターフォール(滝)なのです。 多くの開発は半年～1年以上の期間がかかるため Web サービス利用者の声を開発に反映するのに時間がかかる（細かく反映することには向かない）という弱点があります。

なお、ウォーターフォール開発は悪ではありません。今後も消滅はしないと思われます。 開発手法の違いはシステム開発に求める価値観の違いに依存するのです。

【WF 弱点 2】 見積りが正しくないのに納期固定の完成責任が発生するという本質的な矛盾から、開発担当者のモチベーションが低下することがよくあります。 その場合、良いソフトウェアが作られません。

ウォーターフォール開発には、もっと大きな問題点があります。 それは正しくない見積りに対して、請負開発契約での納期固定で完成責任が発生する本質的な矛盾です。 開発プロジェクトがデスマーチ（死の行進）化すると、開発担当者のモチベーションが低下し、そのとき作成されるソフトウェアは仕様を満たす最低限のものになります。

別のケースとして、ウォーターフォール開発を前提としながら仕様の FIX をせずに開発を実施することで、ソフトウェア開発プロジェクトがデスマーチ化することもあります。 この場合も同様に、モチベーションが低下した開発担当者によって、最低限のソフトウェアが作成されます。

これらの課題に正面からウォーターフォール開発で対応するには、高精度の再見積りと機敏な契約変更の実施が必要です。 しかし、再見積りについても KKD 法であることは変わりませんし、請負契約文化の中では、変更理由の妥当性を経営幹部に理解してもらうことは困難でしょう。 増額や納期延伸も拒否されるかもしれません。柔軟な契約変更という幻想は日本では十分に機能していないと思います。

3K（きつい、厳しい、帰れない？）の開発現場からは、良いソフトウェアは生まれません。 そんなソフトウェアによるサービスでグローバルに他社と対等な競争ができる訳がありません。

1.4 日本の慣習で、なぜアジャイル開発は失敗するのか？

このように優位性のあるアジャイル開発ですが、日本で実施すると失敗することがあります。 それは、なぜでしょうか？

■ 請負開発契約でアジャイル開発を実施すると、良いソフトウェアは作れません。

開発契約には大きく2種類があります。 製造請負契約と準委任契約です。 図 1-1 にその違いを示します。

アジャイル開発では、発注側の PO が いつでも仕様の変更の権利を持ちます。 よって、請負開発ッまり**金額固定の請負である仕様 FIX の合意がありません** 請負開発契約によってアジャイル開発を実施することは、受注側がどんなに精度高く見積っても全く意味の無い見積りでソフトウェア開発を実施することに同じです。 これでは契約を超えた特別な調整でも無い限り、良いソフトウェアが作成されることはありません。 そんな中で製造されたソフトウェアに対して品質保証(瑕疵担保責任)ができるはずもありません。

	標準的な製造請負契約	標準的な準委任契約
見積り	仕様で規定されたものを期限までに完成させる対価	作業者の単金(単価)（時間単金や経過時間時間時間単価等）
完成責任	受注側	なし（発注側にあり）
瑕疵担保責任（代金減額等）	受注側にあり	（特約で付加可能）
善管注意義務（善良な管理者としての注意義務）	―	受注側にあり
支払い時期	完成後	途中請求可能
第三社再委託	原則自由（特約で事前承諾あり）	なし（特約で第三者発注可能）
検収規定	受入試験・検収合格	稼動時間報告や作業報告
作業管理責任者	必要（ご注文や指示が出来ない場合は管理者が作れないかもしれない）	
発注社リスク	仕様書が曖昧だったための期待と違うものが納品された。仕様変更に対応できず、完成時には時代遅れ	発注側指示の失敗による未完成や低品質
仕様変更	契約変更（金額・納期の見直しが必要）	契約内のいつでも可能

※2017年の民法(債権法)改正によって2020年4月から準委任契約でも成果に対して報酬を払う契約も可能となる。他にも報告(用語、責任追求期間、修補請求権)について変更がある。

図 1-1 製造請負契約と準委任契約