製造業に学び、越えたアジャイル開発

《IT企業・IT部門の
トップマネジメント・セミナー》

「ソフト開発プロセス改革・
TPS/リーン/アジャイル」

ESD21

1. 製造業の視点からアジャイル開発をとらえ直す
2. アジャイルなSIになるには



2012年3月16日

名古屋市 東桜会館

山本 康彦

BluewaterSoft

自己紹介

BluewaterSoft

Microsoft BizSpark Start up

山本 康彦

個人でやってます

BluewaterSoft

http://www.bluewatersoft.jp/

- 名古屋大学工学部(修士)
- HONDA R&Dで自動車設計
- 1994～ ソフトウェア業界
- 2012 独立

著作



- 著書
  - 「速攻入門 C#」(2012/3) 技術評論社、共著
  - 「マルチスレッドC#」(仮) (執筆中)
- 記事
  - 連載 「C#でTDD入門」 翔泳社
  - 「Metroスタイルアプリの開発者が知るべき3つのこと」 デジタルアドバンテージ …etc.

ソフトウェア開発

Windows 8 Metro Style App.
Windows Phone 7
…Windows系の最新技術

TDD(コーディング技法)の普及

まず始めに。

異業種から学ぶために必要な事は?

ことば

経理部
手形が、
「落ちた!」

クリーニング
「落ちた!」

雑誌の編集者
「落ちた!」

プログラマー
「落ちた!」

証券会社の
ディーラー
「落ちた!」

- 異業種との意思疎通は、
  言葉の意味を合わせることから。
- 製造業から学ぶには?
  最初の質問
  「『**製造**』ってなんですか?」

製造?

「品物をつくること。原料に人工を加えて製品とすること。」 ― 広辞苑 第四版

「①〔工場などで大量に〕品物をつくること。
②〔経済〕原料や未完成品を加工して商品をつくること。」 ― 学研国語大辞典

- 辞書的な意味
  - 品物を作る
  - おもに大量生産を言う
- 製造業での認識
  「**売り物を作るのが製造**」
- 言葉を使い分けている
  - **製造** ： お客様に渡せるモノを作る
  - **試作** ： 開発中に試しにモノを作る
    (お客様には引き渡せない)

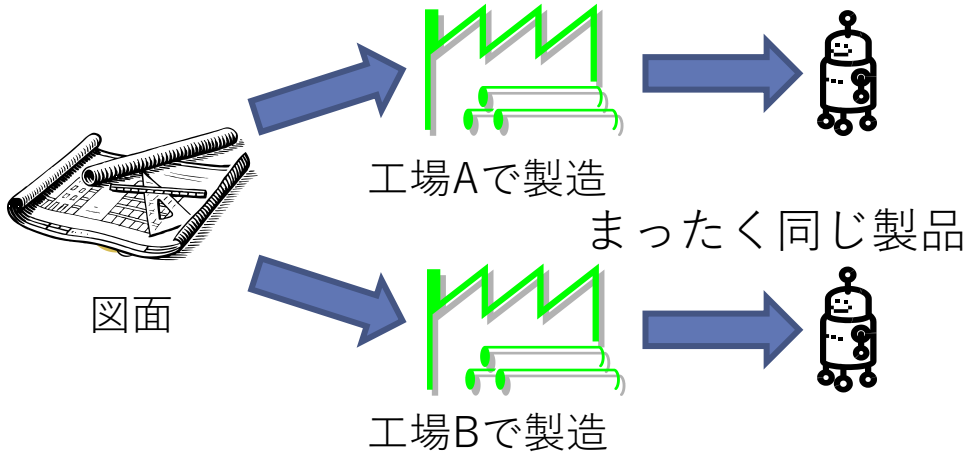
製造?

もうひとつ条件が

- 同じモノが作れる
  - どこの工場でも
  - 誰が作っても
    …何を与えれば?

＝お客様に渡せるモノを作る

どこで?
⇒ 工場で。



製造 ＝ 図面から製品を作る

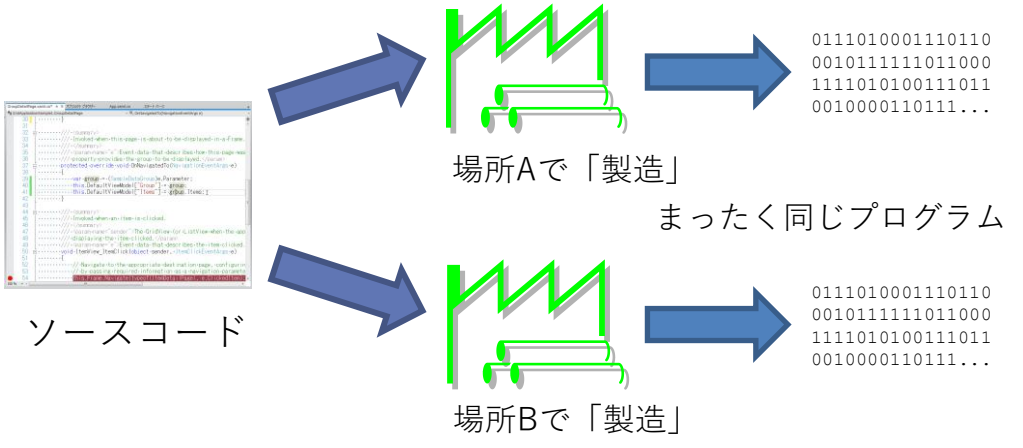
製造?

製造業に合わせて考える

- 同じプログラムが作れる?
  - どこで作っても
  - 誰が作っても
    …何を与えれば?

ソフトウェアでは?

製品は?
⇒ プログラム



製造 ＝ コンパイル/ビルド

- ~~製造~~ ＝ 仕様書からソースコードを作る
- それは、製造業での定義を満たすか?
  - どこで作っても
  - 誰が作っても
    まったく同じソースコードが出来上がるか?
- 答えは No!
  仕様書は曖昧で、組織によって・人によって、違うソースコードになってしまう。

同じモノが作れないなら、製造業的には「**製造**」**ではない!**

ことばが一致すると、

いままで気づけなかったことが
見えてくる。

製造

- 製造業の「製造」に当たるのは、コンパイル/ビルド
- それはマウスクリック一発で…

自動で行われる

＝コンパイル/ビルド

それって…
誰がやってる?

ソフトウェア業界では、
すでに前世紀から

製造は100%自動化済み!!

試作さえも自動化済み!
じつは製造業より遥かに進んでいる

では、本題。

製造業とソフトウェア産業、モノ作りの過程を比較

モノ作り過程

- モノ作りの流れに、根本的な違いがあるわけではない。
- ただし製造業には、「どうやって量産するか」(工程エンジニアリング)がある。

※ 製造業での区分に合わせて、ウォータフォールを再配置した。

製造業

コンセプト → 製品基本計画 → 製品エンジニアリング →  図面 → 工程エンジニアリング → 製造 → 出荷 

WHAT? HOW?

ソフトウェア

コンセプト → 要件定義・外部設計 → 内部設計・コーディング・テスト →  ソースコード → 最終ビルド → リリース

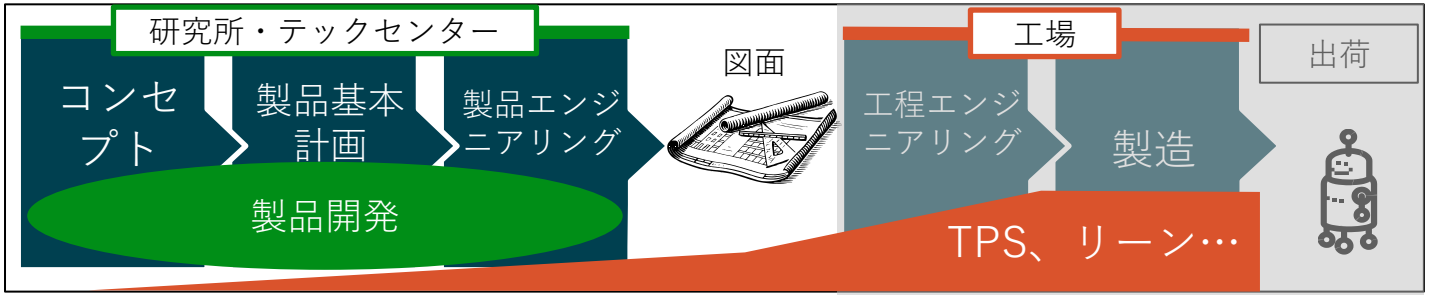
```
01110100
01010110
00101111
11011000
10111...
```

何処から学ぶ?

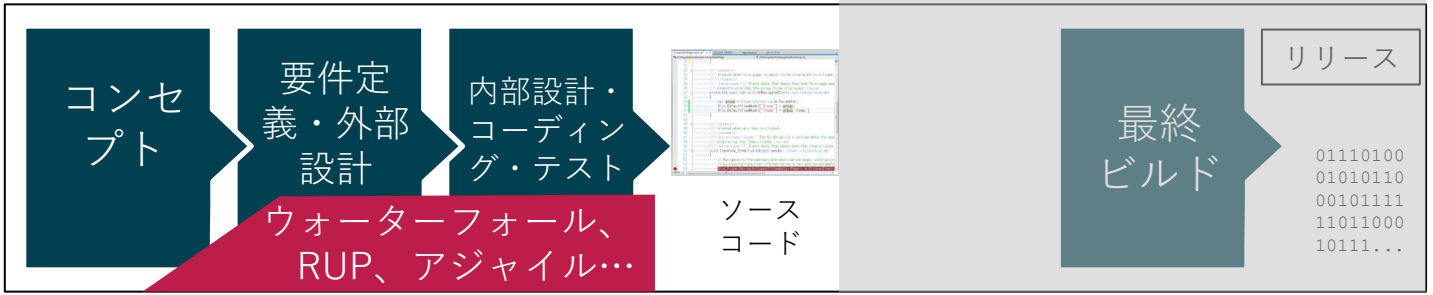
- ソフトウェアの開発プロセスに当たる部分は、製品開発（図面が完成するまで）

TPSやリーン生産方式の主な対象は、製造。
ソフト開発に直接は適用できないので、注意。

製造業



ソフトウェア



製造業の製品開発から学ぼう!

ところが…

製品開発プロセスは社外秘

おいそれとは見せてもらえない

製品開発

- 機密事項なのでなかなか知ることは難しいが、守秘義務契約の元に書かれた記事や論文がある。東大の藤本教授が著名。

守秘義務があるので、現役・退職者が公の場で許可無く語ることは無い。

記事



「プロダクト・インテグリティ すり合わせの製品開発力　―日欧米自動車メーカー20社への調査が明かす―」藤本 隆宏、キム･B･クラーク(著)
「ダイヤモンド・ハーバード・ビジネス」1991年 2－3月号
※「増補版 製品開発力」(2009)にも同内容が含まれる

論文



「自動車製品開発のプロセスと組織」藤本隆宏
東京大学COE ものづくり経営研究センター MMRC DISCUSSION PAPER No.61 2006年1月（前半はインターネットで閲覧可能）
※次の説明する製品開発フローは、この論文による。

製品開発

- 製品エンジニアリングでは、詳細設計/製図→試作→実験/評価というループを、計画に従って数回繰り返す。

Why?
機械は複雑なので、一回で間違いなく設計することは無理。

コンセプト創造

どんなものが欲しいか

製品基本計画

WHAT?
具体的にどんなものか

- クレイモデル
- レイアウト図
- 要求仕様/目標（スペック）
- 部品技術選択

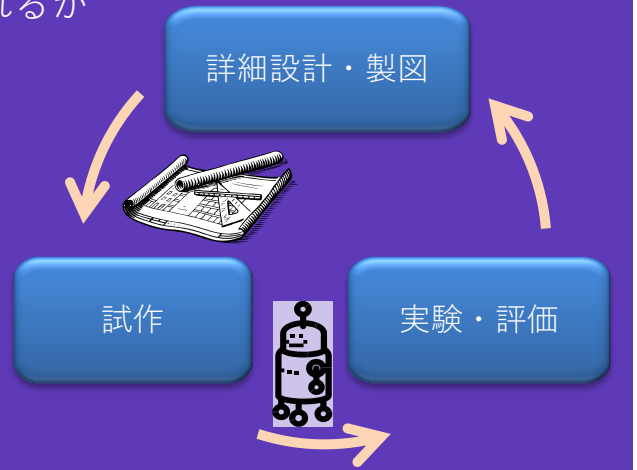
…etc.

製品エンジニアリング

HOW? どうやったら作れるか

※ 詳細設計:
1. 部品のスペックを決める
↓
2. スペックを実現する方法の算出・考案

※ 「スペック」は、テスト可能。
テスト方法と合否基準を決める。



まねてみる

- 製造業では、図面を描く前に、スペックを決める。
- ソフト開発でも、テストケースを先に決めたらどうか?

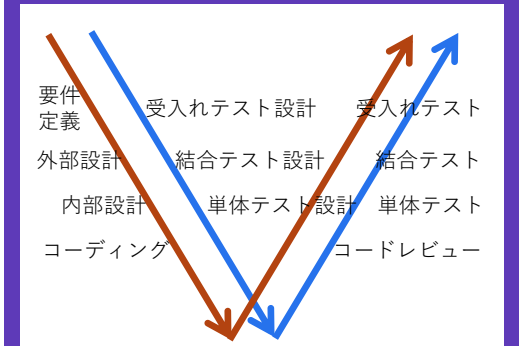
すでに実践しているところも多い

製造業では、
スペック決め ⇒ 作図

ソフトに置き換えると、
テストケース決め
⇒コーディング
…ということに。

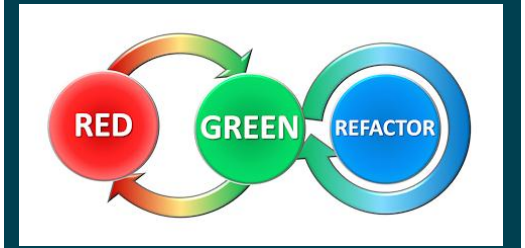
W字モデル

次工程に進む前にテスト仕様書を作る



TDD(テスト駆動開発)

自動化されたテストケースを先に書く



まねてみる?

- 製造業では、繰り返しの回数を減らそうとしてきている。
- ウォーターフォールでは、実は無定見に繰り返している。ここは改善すべき?

?

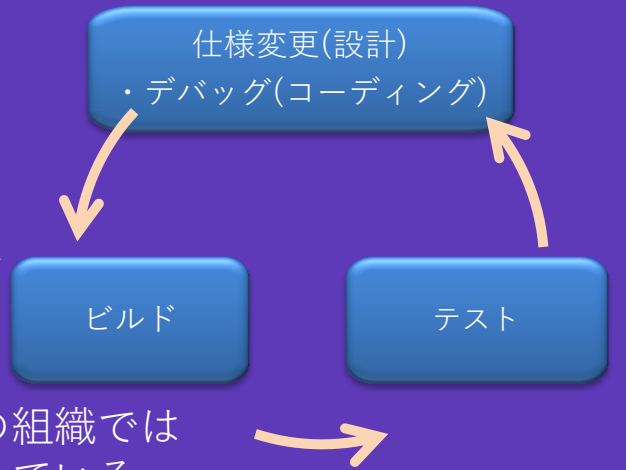
製造業では、
製品エンジニアリングを
繰り返す

ソフトに置き換えると、
内部設計・コーディング
・テストを繰り返す
ということだが…。

「ウォーターフォールは繰り返しをしない」神話

結合テスト工程の
中を見てみる

テストで不具合発見
⇒ 仕様変更/デバッグ
⇒ ビルド
⇒ テスト
…このループを多くの組織では
てんでバラバラに回している



ループの特性

- 設計⇒試作⇒テストのループは、製造業とソフトウェアでは、反対の性質を持っている。

製造業は、ループ回数を減らすために、シミュレーション技術を導入している

製造業

試作
コスト・時間が掛かる。自動化しにくい。

テスト
コスト・時間が掛かる。自動化も高コスト。

なんとかして減らしたい!

ソフト

試作
100%自動化済み。時間も数秒から数分程度。

コンピューティング パワーを活用

テスト
大幅に自動化可能。TDDでは実装工数2割増し程度。

どれだけ繰り返しても平気になった

試作品の特性

- 製造業では完成品にしないと機能テストができない。
- ソフトでは、機能ごとにでも、作ってテストできる。

自動車の機能は「走る・曲がる・止まる」。曲がる機能だけ作ってテストできるだろうか?

製造業

設計⇒試作
コンポーネントごとには作れる。機能単位ではムリ。

テスト
機能単位でテストしたいがムリ。完成品が基本。

完成するまで機能テストはできない

ソフト

設計⇒試作
コンポーネントごとでも、機能単位でも作れる。

カタチのないソフトならでは。

テスト
完成品になっていなくても、機能単位でテスト可能。

機能ごとに試作してテストできる

製造業と比べると、ソフト開発では、

・繰り返しが苦にならない※

・機能ごとに作ってテストできる

※自動化テストが前提

これらの特性を生かした
開発プロセスはどうなるだろう?

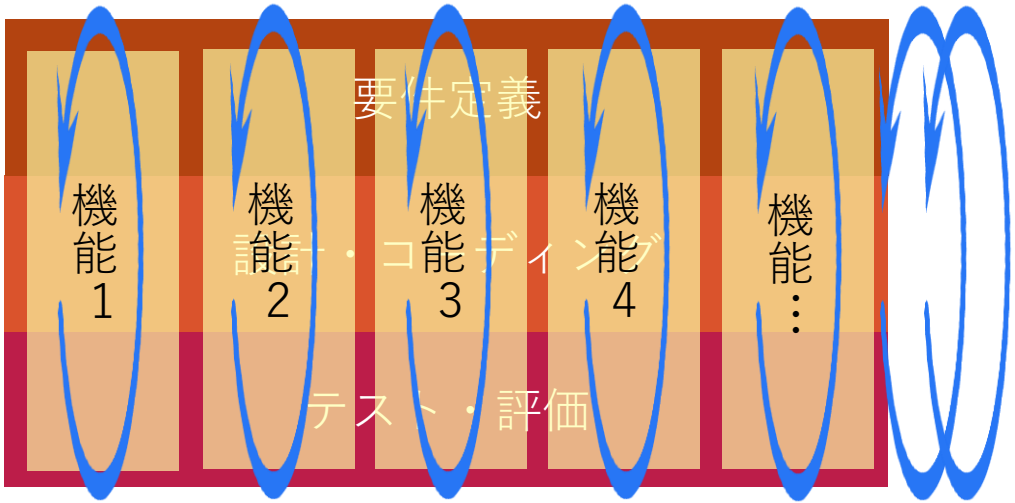
製造業にはできない方法

- ソフトでは、機能ごとに繰り返し作っていける。
- 明確な機能から始めて、だんだんと育てていける。

インクリメンタル&イテレーティブな手法

繰り返す

機能ごとに



全体を繰り返すのは、製造業でやっている方法

小刻みに機能単位で繰り返すことは、製造業では真似できない

ビルドとテストの自動化を
バックボーンとして

インクリメンタル&イテレーティブな
開発が可能



アジャイル開発が可能に!

念の為…

インクリメンタル
&イテレーティブ

≠

アジャイル開発

その違いは…?

アジャイル宣言

http://agilemanifesto.org/iso/ja/

- インクリメンタル&イテレーティブなプロセスであり、**アジャイル宣言にコミットしている**のが、アジャイル ソフトウェア開発

Manifesto for Agile Software Development

宣言（4つの価値観）

私たちは以下の価値に至った。

・プロセスやツール　　よりも　個人と対話を、
・包括的なドキュメント　よりも　動くソフトウェアを、
・契約交渉　　よりも　顧客との協調を、
・計画に従うこと　　よりも　変化への対応を、

価値とする。すなわち、左記のことがらに価値があることを認めながらも、私たちは右記のことがらにより価値をおく。

12の原則

1.顧客満足を最優先し、価値のあるソフトウェアを早く継続的に提供します。/2.要求の変更はたとえ開発の後期であっても歓迎します。変化を味方につけることによって、お客様の競争力を引き上げます。/3.動くソフトウェアを、2-3週間から2-3ヶ月というできるだけ短い時間間隔でリリースします。/4.ビジネス側の人と開発者は、プロジェクトを通して日々一緒に働かなければなりません。/5.意欲に満ちた人々を集めてプロジェクトを構成します。環境と支援を与え仕事が無事終わるまで彼らを信頼します。/6.情報を伝えるもっとも効率的で効果的な方法はフェイス・トゥ・フェイスで話をすることです。/7.動くソフトウェアこそが進捗の最も重要な尺度です。/8.アジャイル・プロセスは持続可能な開発を促進します。一定のペースを継続的に維持できるようにしなければなりません。/9.技術的卓越性と優れた設計に対する不断の注意が機敏さを高めます。/10.シンプルさ（ムダなく作れる量を最大限にすること）が本質です。/11.最良のアーキテクチャ・要求・設計は、自己組織的なチームから生み出されます。/12.チームがもっと効率を高めることができるかを定期的に振り返り、それに基づいて自分たちのやり方を最適に調整します。

アジャイルという言葉

- "Agile Software Development" という言葉は、アジャイル宣言を策定した会議で決められた。
- アジャイル宣言を無視してアジャイル開発を説明することは、できない。

Manifesto for Agile Software Development

宣言以前

開発プロセスに対して、「アジャイル開発」という呼び方は無かった。

宣言策定会議

2001年2月11～13日 米国ユタ州スノーバードに、17人の開発者が会議。
our first task was to come up with a new adjective that we could live with. Now our processes are "agile", … 「**我々の最初の仕事は、全員が受け入れられる新しい形容詞を考え出すことだった。**今、私たちのプロセスは『アジャイル』である」
http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf
アジャイル宣言のために、「アジャイル開発」という言葉を作った。

アジャイル開発をするには?

- アジャイルな開発チームとアジャイルな顧客が、車の両輪。

そして、それをサポートするマネジメント

アジャイルな開発チーム

- アジャイル宣言の4つの価値にコミットし、12の原則を守るべく行動しているチーム。
- 自己組織的であり、振り返りを行って常に改善し続けているチーム。

アジャイルな顧客

アジャイル宣言の4つの価値にコミットし、12の原則を尊重する顧客。

顧客がアジャイルでないと
チームが原則に従えない ⇒アジャイル開発ではない

ここまでのまとめ
製造業に真似できない、アジャイル開発

・バックボーン: コンピューティングパワーを活用した試作とテストの自動化
・アジャイル宣言の4つの価値と12の原則
・アジャイルなチームと、アジャイルな顧客

アジャイルな契約

- アジャイル開発では、従来の契約形態ではうまくない
- 顧客企業が、プロジェクトのために開発者を雇い入れる欧米型では問題が少ないが…

マネジメントの課題

アジャイルはスコープが変動する

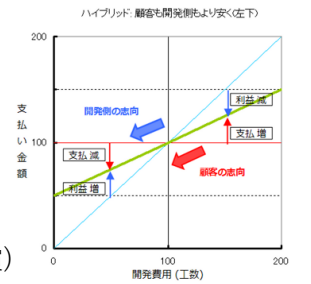
- 固定金額
  顧客:増やそうvsベンダー:減らそう
- 工数比例金額
  顧客:減らそうvsベンダー:増やそう

いずれにせよ、対立する

アジャイルに向いた契約

試行錯誤が必要
顧客とベンダーがともに納得できる契約形態を探らねばならない

【例】BestBrains社のハイブリッド契約方式
(2010年、ESD21の招待講演)



アジャイルな人事考課

- アジャイル開発では、従来の人事考課も上手く機能しないことがある

マネジメントの課題

アジャイルはチームで働く

- 個人の生産性は測定できない
  コードの複数所有: あるコードを複数人で書いているから、一人あたりのコーディング量は測れない。バグも同様。

アジャイルは一人何役もこなす

- 役割によってランク付けできない
  従来の工程で言う要件定義/設計/実装/テストを、チームの誰もがやる。要件定義をやるから上級SEといったようなランク付けは通用しない。

アジャイルになるには?

- ここまで説明してきた4つの課題が重要。
- アジャイル宣言の4つの価値と12の原則を忘れずに。

マネジメントの課題

対内	アジャイルな**開発チーム**を育てる	アジャイルに適した**人事考課**を探る
対外	**アジャイルな顧客**を探す、育てる	アジャイルに適した**契約形態**を探る

…アジャイル開発は、自社ではムリ!?

・「うちでは、とてもアジャイル宣言の4つの価値と12の原則を、満たせない…。」

・⇒ アジャイル開発の良いところを、取り入れていきましょう!
　アジャイル型、アジャイル風

※ 会話ではどれも「アジャイル」と言います。要注意!!

ほんとうに大切な事

- 開発プロセスを改善し続けること
- アジャイル開発、アジャイル型、アジャイル風
…なんだって構わない!

カイゼン

…閑話休題

TDD（テスト駆動開発）

関数を作る

- TDDは、「テスト駆動**開発**」という名前が付いているが、関数を作るための**技法**。

Test Driven Development

前世紀のやり方

- 1) **外部設計書**を書く。関数の入出力や状態遷移など。
- 2) **内部設計書**を書く。フローチャートなど。
- 3) **コーディング**する。
- 4) 外部設計書に基づいて、**単体テスト**を実施する。

あれ!? たいして違わない

TDD

- 1) 外部設計を、**自動実行できるテストケース**として書く。
- 2) 製品コードを**コーディング**する。
- 3) **自動化されたテスト**を実施する。
- 4) **内部設計を改良**する。(リファクタリング)

TDD
どこが違う？

- インクリメンタル&イテレーティブ
- コンピューティング パワーの有効活用

Test Driven Development

小刻みに繰り返す

インクリメンタル

たとえば入出力表の同値クラスごとに、テストケース⇒コーディング⇒テストを繰り返す

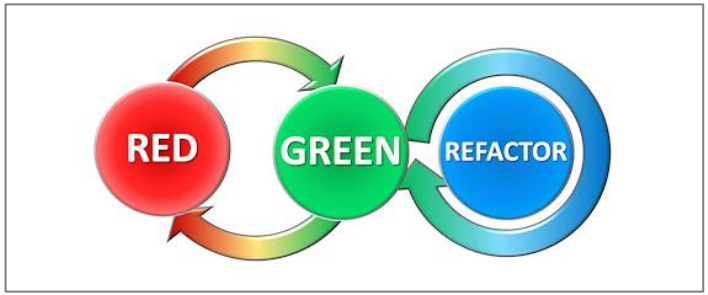
イテレーティブ

一回で良いコードを書くのはムリだから、繰り返し改良する

演算能力有効活用

ビルド/テスト時間の短縮

関数ひとつに何十回も何百回も繰り返す。演算能力が上がって、初めて可能になった。



アジャイル開発、TDD 共通の特徴

- インクリメンタル [小刻み]:
  立ち向かう課題を小さくする
- イテレーティブ [繰り返し]:
  一度で完璧にはできないから、繰り返す
- コンピューティング パワーの活用:
  以前はできなかった

製造業に学び、越えたアジャイル開発

ご清聴 ありがとうございました