最速でJavaエンジニアになって稼ぐ

2018/11/24

人生逃げ切りオンラインサロン内

目次的なやつ

- 開発手法(ウォーターフォール、アジャイル)
- 開発手法(テスト駆動開発、チケット駆動開発)
- テスト手法とテストの種類
- 生産性を上げる為の具体的な方法
- フレームワークの知識(Spring、Struts、JUnit、Selenium)

開発手法
(ウォーターフォール、アジャイル)

ウォーターフォールとは名の通り
水が流れるように上流から下流へ工程が
遷移していく様を表現した開発モデル。

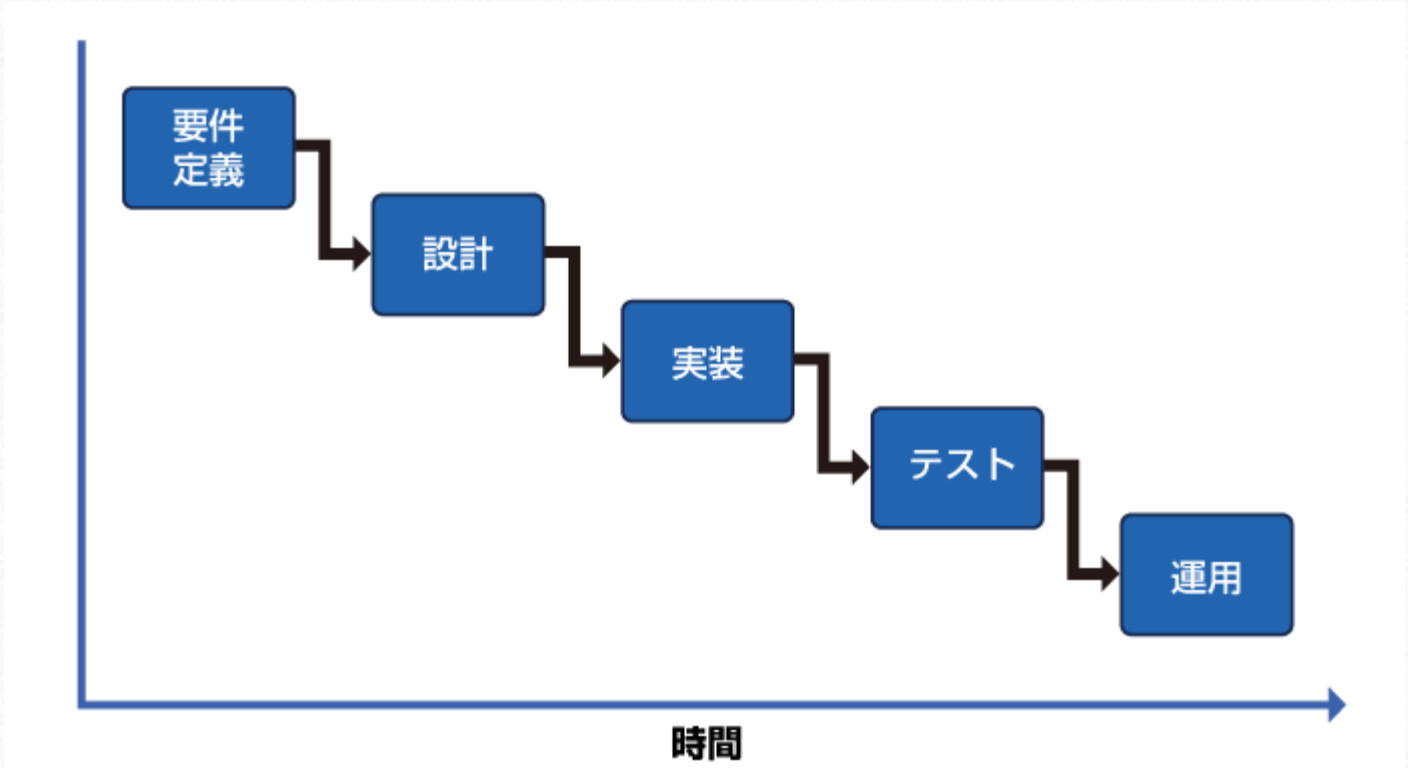
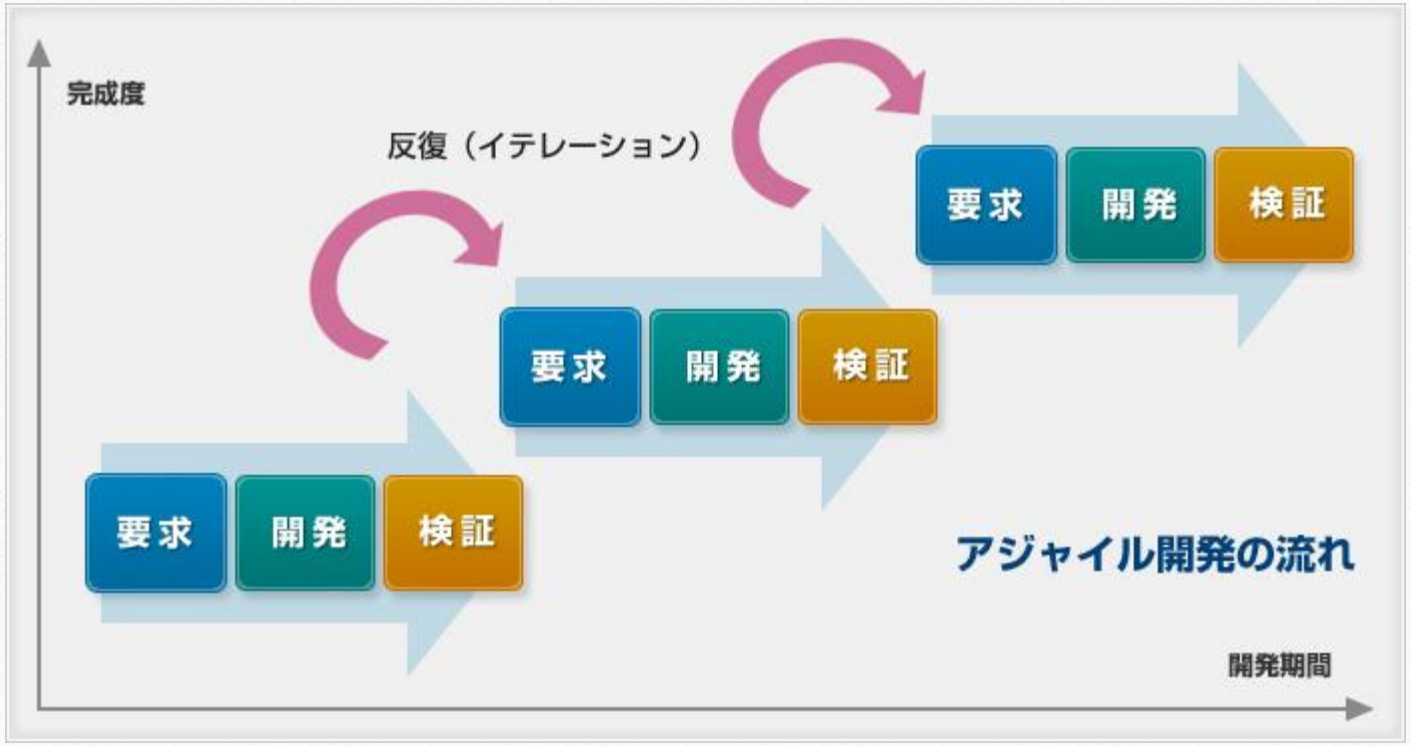


図 1：ウォーターフォール・モデル

アジャイルとは、イテレーションと呼ばれる短い開発単位を反復して繰り返す事で「素早く」「リスクを低く」開発するモデル。

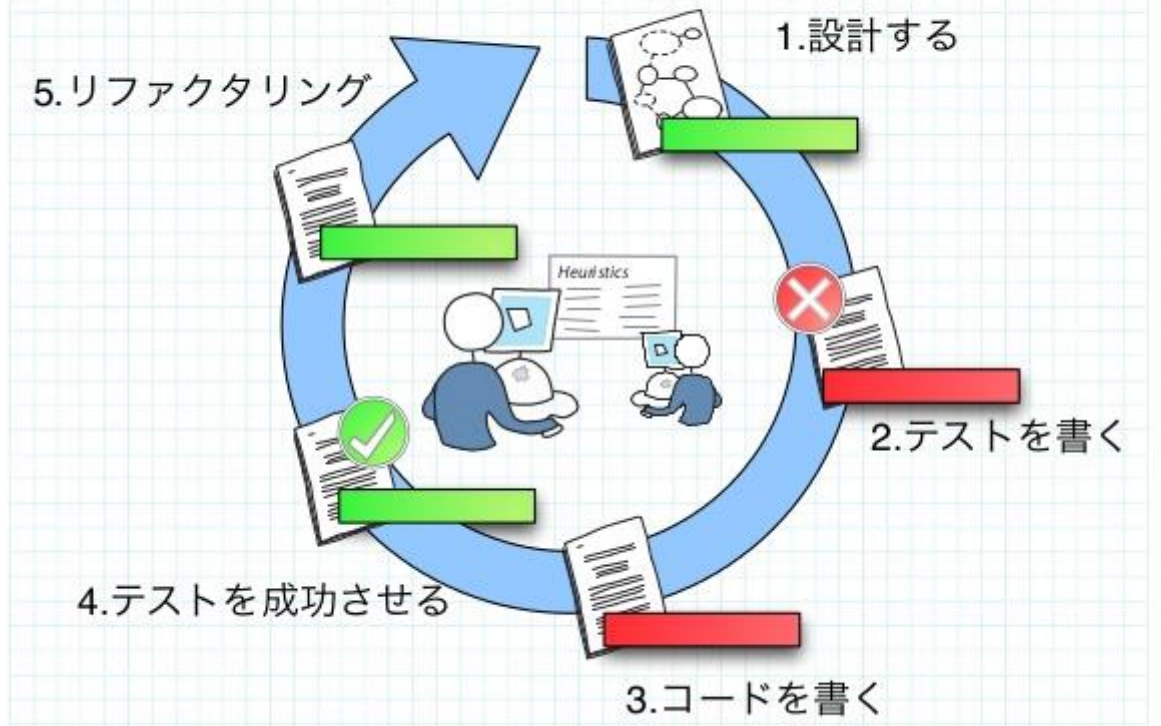

完成度
反復（イテレーション）
要求
開発
検証
要求
開発
検証
要求
開発
検証
アジャイル開発の流れ
開発期間

アジャイルについて語る際、良く話に出てくるのが「スクラム」と「XP」
今回は解説しないけど、調べておく価値有。

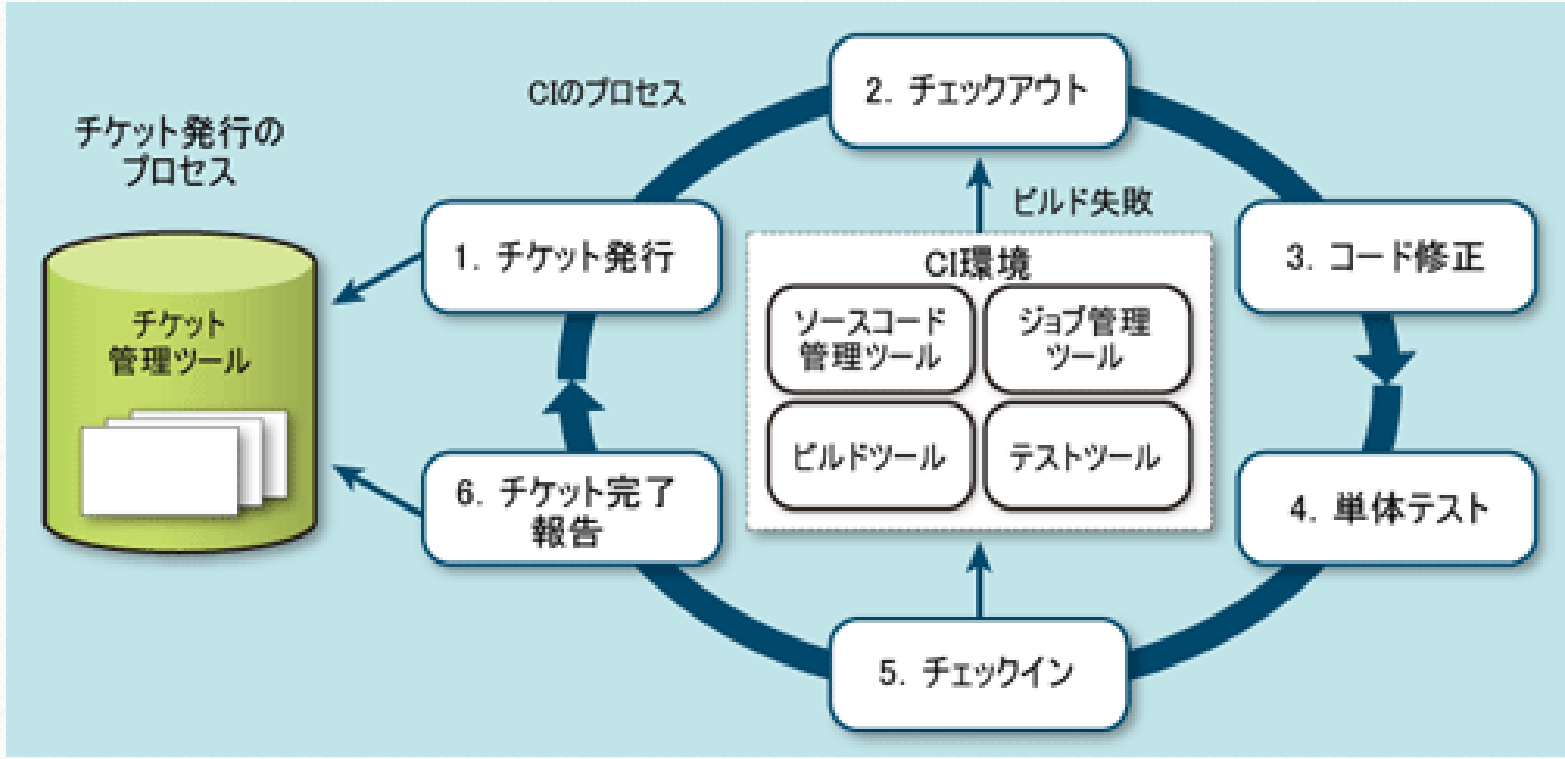
開発手法
（テスト駆動開発、チケット駆動開発）

テスト駆動開発は、その機能(テスト対象)が満たしておくべきものを列挙し、先にテストコードを書き、そのテストが通るようにプロダクトコード作成していく開発方法。

テスト駆動開発のサイクル



チケット駆動開発は、チケットと呼ばれるタスクの単位を設定し、そのタスク単位で作業(開発)を進めていく開発方法。


CIのプロセス
チケット発行の
プロセス
2. チェックアウト
ビルド失敗
1. チケット発行
CI環境
3. コード修正
チケット
管理ツール
ソースコード
管理ツール
ジョブ管理
ツール
ビルドツール
テストツール
6. チケット完了
報告
4. 単体テスト
5. チェックイン

テスト手法とテストの種類

まずはテストの種類から

単体テスト

クラス・機能・処理・メソッドといった比較的
細かい粒度で行うテスト。
各社・各プロジェクト毎に定義づけされる。

結合テスト

クラスがいくつか纏まって「機能」が作られる。
その機能確認のテスト。
　但しこれも定義が分かれ、場合によっては
「機能」と「機能」の連結確認のテストにもなる。

画面遷移のテストなども結合テストに含む。
単体・結合テストまで終わらせて納品というケースもある。

総合(システム)テスト

「機能」が纏まって「システム」として動作するかを確認する為のテスト。
動作確認としてのテストは結合までで、総合としては性能確認・負荷確認・セキュリティ・ユーザビリティなどのテストが行われる。

規模が大きい場合、システム同士の結合確認もあるため、総合テストの中で「シナリオテスト」と言う業務処理を模した形でのテストも行う。

受入テスト

納品されたシステムや機能を
お客さん(ユーザー)が確認するテスト。
機能要件や運用面の確認をするテスト。
これが通らないと納品した事にならない。

テスト手法のあれこれ

ホワイトボックステスト

システムや機能の中身がわかった上でのテスト。
例えばif文による分岐処理のテストや
値の限界値のテストなど。
作成者であれば知っている事をテストする。

ブラックボックステスト

要件や求められている機能が実現されているかを確認する為のテスト。
中身がどうかは意識せず、要件・仕様に注力する。
テストで用いる値も実際に利用されるような値を用いる。

生産性を上げる為の具体的な方法

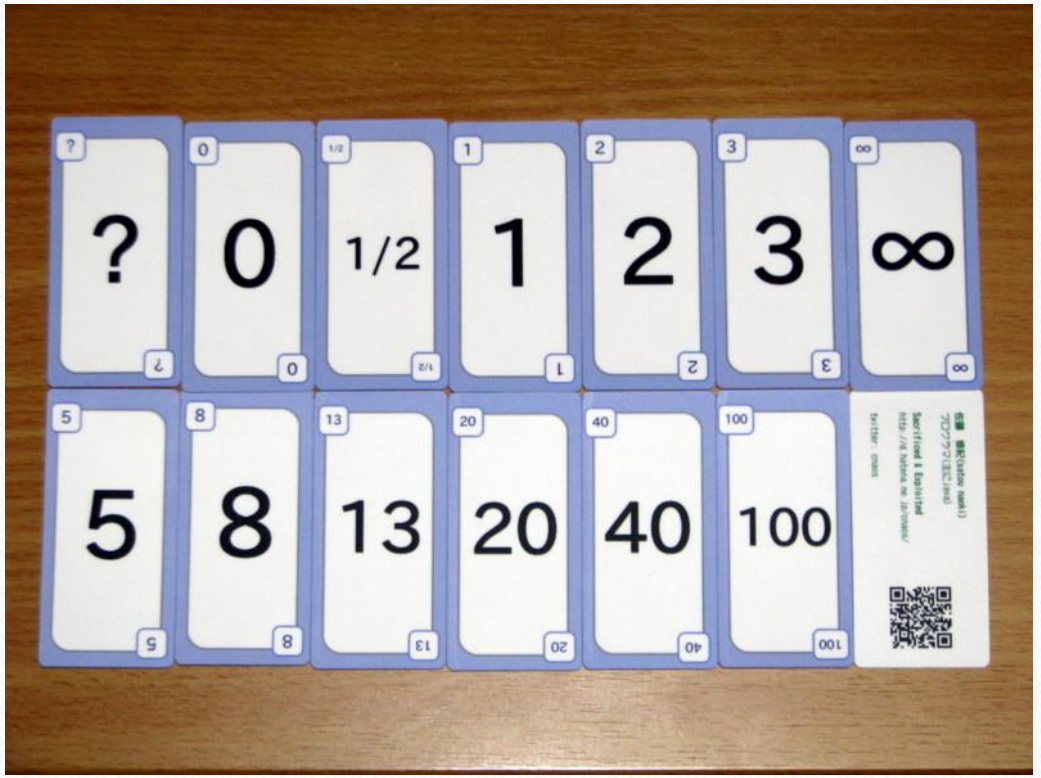
カンバン

やる事・やってる事・終わった事。
といったような区分を作り、タスク管理する方法。
1人でも出来る。


普通のカンバン
非公開
TODO
XXX API クライアントの実装
YYY 一覧画面の実装
ZZZ APIの実装
カードを追加
Doing
AAA アイコンの変更
B にコメント投稿
C に写真添付する
カードを追加
Review
XXX クラッシュの対応
YYY テーブルの設計
カードを追加
Done
XXX 詳細画面で動画プレイヤーの実装
カードを追加

プランニングポーカー

チーム開発を行う際、メンバー間でチケットなどのタスクに対する見積りを行う方法。
各それぞれで工数見積りを行い、一斉に見せて、差分が有る場合に議論を行い、
見積りずれをなくす。


?
0
1/2
1
2
3
∞
5
8
13
20
40
100

振り返り(KPT)

Keep(やった事や維持したい事)
Problem(問題があった事)
Try(チャレンジ・改善したい事)

の単位で任意の開発期間を振り返る。
振り返る事で改善を出し、より良くしていく方法。
アジャイルやスクラムなどのイテレーション単位で行われることが多い。

実施例

フレームワークの知識
(Spring、Struts、JUnit、Selenium)

SpringFramework

Springは極端に言うと「なんでも出来てしまうフレームワーク」
様々なモジュール群の集まりで構成されており、それぞれ個別で利用可能。

DIコンテナ・トランザクション管理・データアクセス・認証認可・バッチ処理・MVC・テスト・アスペクト指向などなど。

とりあえずSpring使っておけば、大体の事は出来る。
しかも高水準で出来てしまうので、他のをあえて使う理由がない程。
バージョンアップも早いので、Javaのリリースサイクルにも合わせてくる。強い。

Struts

一世を風靡したある意味Javaを流行らせたフレームワークがApacheStruts。
Struts2というものが現在は最新で、
Struts1とは似ても似つかない。

Struts1は当時非常に扱いにくかった
Servlet/JSPの機能拡充したような仕組
みが人気の要因。

しかし現在においては脆弱性や新しいフ
レームワークは技術の台頭についていけ
ず、サポート終了となっている。

Struts2は仕組みを一新して、Springが持っているようなDIコンテナの機能やアノテーションによる設定ファイルの削減などを行ったが、Springでより洗練され、かつ容易に使えてしまう為、流行らなかった。

更に言うとStruts1とも互換性はない上、セキュリティホールが多々見つかっているので、現在において採用理由は薄い。

JUnit

JUnitは古くからあるJava用のテスティングフレームワーク。

プログラムをテストする場合、そのプログラムを動かす必要があるが、それを容易にするためのもの。

非常に簡単に扱えるため、デファクトスタンダードとなっている。

Selenium

ブラウザ上の動作を自動化し、テストする為のテスティングフレームワークの一種。

但し厳密には、FireFoxから提供されていたプラグインであり、それをJavaなどの他言語から利用できるようにしたもの。

Seleniumを直接扱う事は最近は少なく、ラップされたFluentLeniumやSelenideを使う。

更に言うと仕組みの根幹は「WebDriver」と呼ばれるブラウザ側の仕組みであり、これを利用して動作しているのがSelenium。

また以前はブラウザ間で振る舞いが異なり、それを利用者側が吸収する必要があったが「WebDriver」が正式にW3Cの標準に加えられたので、今後は振る舞いの違いを気にしなくて済みそうな雰囲気。

次回までの宿題

SpringBootによるアプリ開発の完了。
出来てない人は個別に相談。

おしまい