今日はシステムを作るエンジニアや開発者の気持ちで話を聞いてみましょう！

テクノロジ 4択45問	ストラテジ 4択35問	マネジメント 4択20問

今日の内容をやったら、マネジメントの足切りは絶対に回避できる！

プロジェクトとは物や目標を、制作・達成すること。
①目標が明確にされていること。
②開始時点と終了時点が明確に決まっていること

M2 Macbook air
https://www.apple.com/jp/mac/

プロジェクトマネジメント

プロジェクト達成するための管理のこと

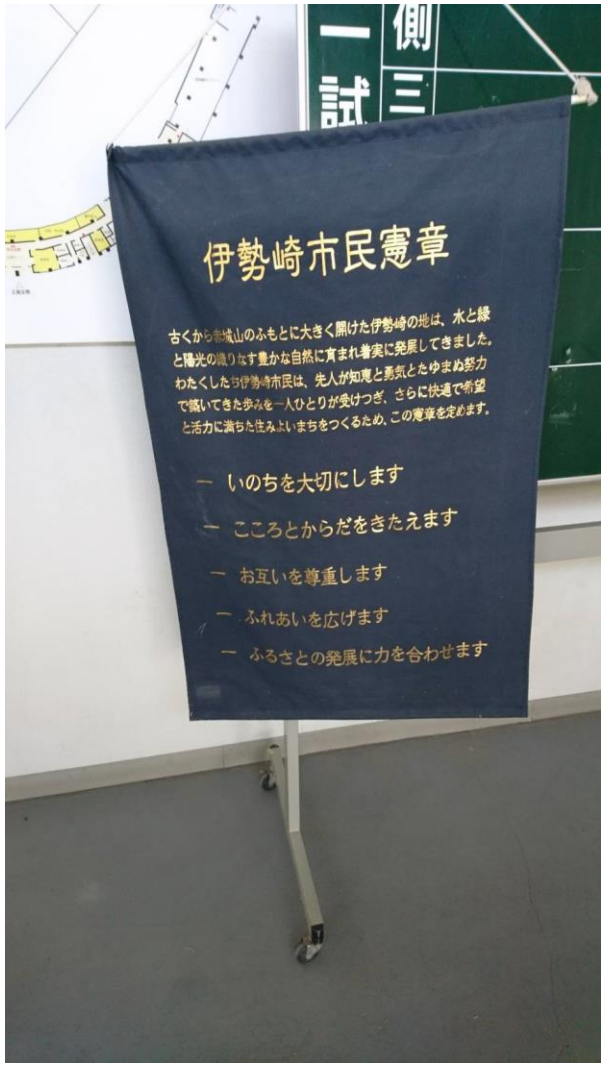
1、目標を立てる。
2、スコープ（目標の範囲や成果物）を定める。

3、スケジュール（WBS）をたてる。
4、人員の配置・体制をきめる。
5、コスト見積り。

6、リスク・品質見積り
7、進捗管理

プロジェクト憲章

【目標を立てる１歩目】
プロジェクトを公布するときの
宣言文
目的や達成物の他は曖昧である
拘束力がない
基本的に修正されない



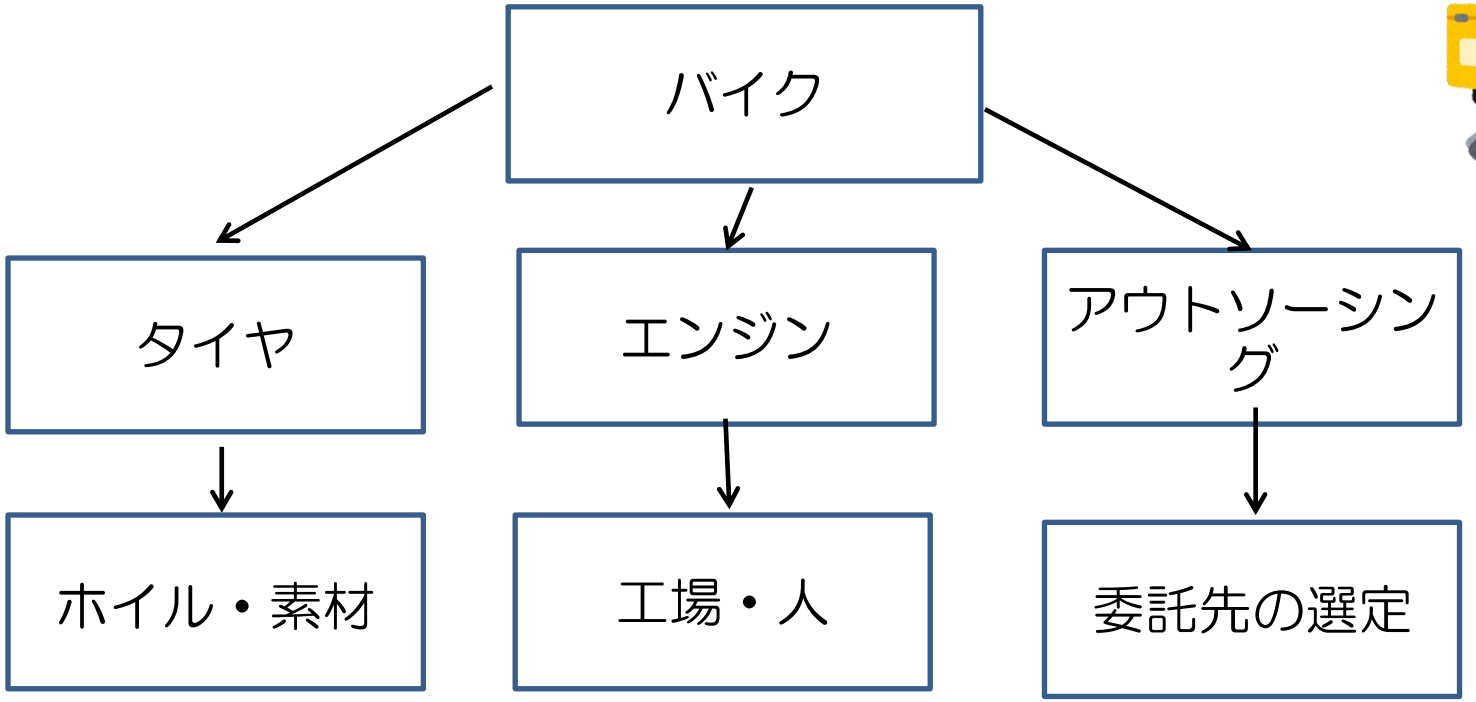
スコープ

【目標を立てる２歩目】
プロジェクト憲章よりも具体的な内容
成果物や法規、顧客の要求を示した物
おおまかな予算やスケジュールも入る

顧客の要求や法規が変更された場合、修正される可能性がある

WBS　（Work Breakdown Structure）

【スケジュール】
作業分解図ともいう。
プロジェクトが決まったら、目標を達成するための作業をトップダウンで分解していく。



WBSのポイント３つ

①レビュー（チェック）をすると
クオリティが上がる。

②要素分解の回数は適度な回数
分解し過ぎると、レビューが増える

③過去のWBSを参考にすると予算は軽減
できるが、クオリティは下がる

レビューって分かる？

WBSはプロジェクトの設計図のようなもの
少しの違いで作業の質が変わるので、チームで
チェックしあうことをレビューという
これにより質が上がる！

WBSのポイント３つ

①レビュー（チェック）をすると
クオリティが上がる。

②要素分解の回数は適度な回数
分解し過ぎると、レビューが増える

③過去のWBSを参考にすると予算は軽減
できるが、クオリティは下がる

要素分解は多すぎず、少なすぎず

やりすぎは何でもよくない
要素分解をすると工程（仕事）が増えてしまう
多すぎても仕事が増え、レビューが増
少なすぎても質が落ちる

WBSのポイント３つ

①レビュー（チェック）をすると
クオリティが上がる。

②要素分解の回数は適度な回数
分解し過ぎると、レビューが増える

③過去のWBSを参考にすると予算は軽減できるが、クオリティは下がる

群馬の企業

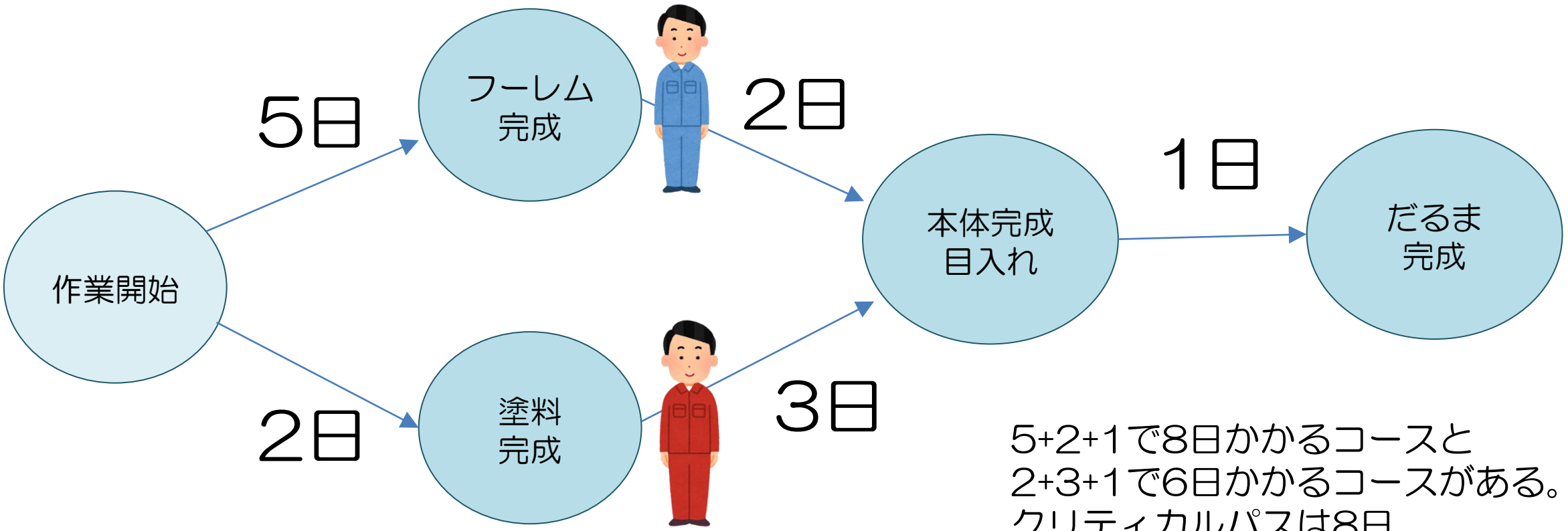
実際の仕事で使うWBSは、詳細な設計図
※成果物、担当者等、**目的を明確**にする
大きいプロジェクトのWBSは～百万とかかる
過去のWBSを再利用して、コストを下げることも

PMのまとめ

家族旅行に行く場合
①プロジェクト憲章
「ハワイに行くぞ」「沖縄に行くぞ」
②スコープ
8月10日～17日まで！予算20万！
③WBS
10日は～～～、11日は～～～

アローダイアグラム

作業日数や順序を表した図
クリティカルパスとは最大かかる日数



5+2+1で8日かかるコースと
2+3+1で6日かかるコースがある。
クリティカルパスは8日

ガントチャート

横軸を時間とし、作業時間を矢印で示した工程管理図のこと
進捗管理に使われることが多い

	2023年				2024年
	1-3月	4-6月	7-9月	10-12月	1-3月
①MOOC完成	→→				
参加者募集	→→				
②ITパスチャレンジ		→→→	→		
③AI検定チャレンジ			→→→	→	
④DS検定チャレンジ				→→→	→
報告書・成果の作成				→	→→→

故障時の対応

ユーザはヘルプ(サービス)デスクに問い合わせをする！

①壊れた商品を問い合わせ

連絡

②サービスデスクが応対

エスカレーション

④上司に対応を委ねる
エスカレーション

確認

③ITILの確認
→成功事例集から対応を検討

確認

③SLAの確認
→サービス内容の合意書
定期的なSLAの見直しをSLMという

システム開発手法とは

規模
製品化
アップデート

１人ではない限り
システム開発手法を
使うことになる

ウォーターフォールモデル

要求定義
→要件定義

外部設計
画面上の遷移

内部設計
アルゴリズム等

上流設計

プログラム
コーディング

テスト
バグ取り

運用保守
問題が起きないように

下流設計

●大規模開発モデル
～年単位の規模
●後戻りが難しい

プロトタイプモデル

●小～中規模開発モデル
数か月単位の規模
●ユーザの意見を反映

再設計

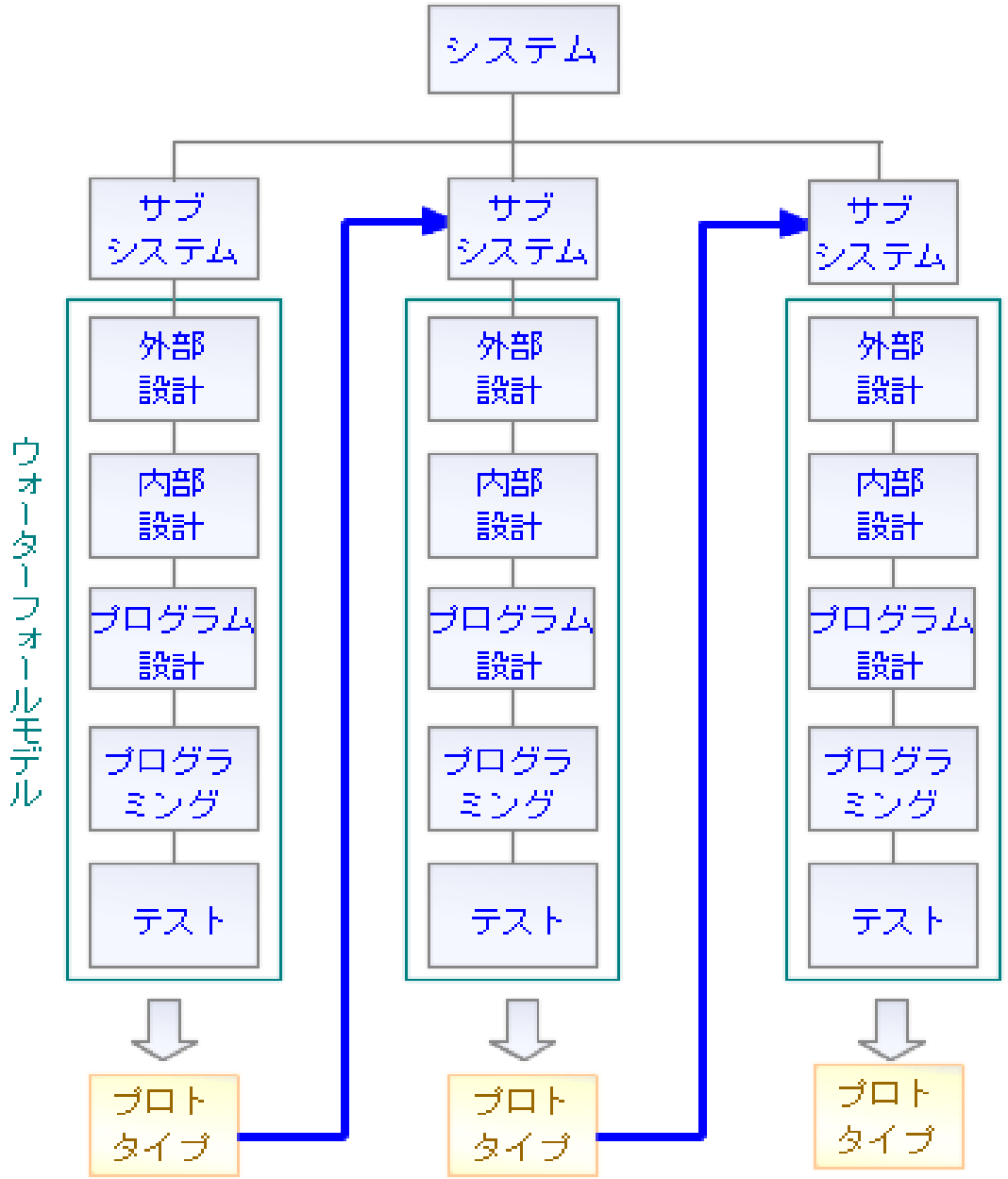
再設計

ユーザ曰く
小さい！
遅い！

ユーザ曰く
タッチパネル
高機能！

試作品をつくり、早い段階からユーザに
批評→改良していくモデル！

スパイラルモデル

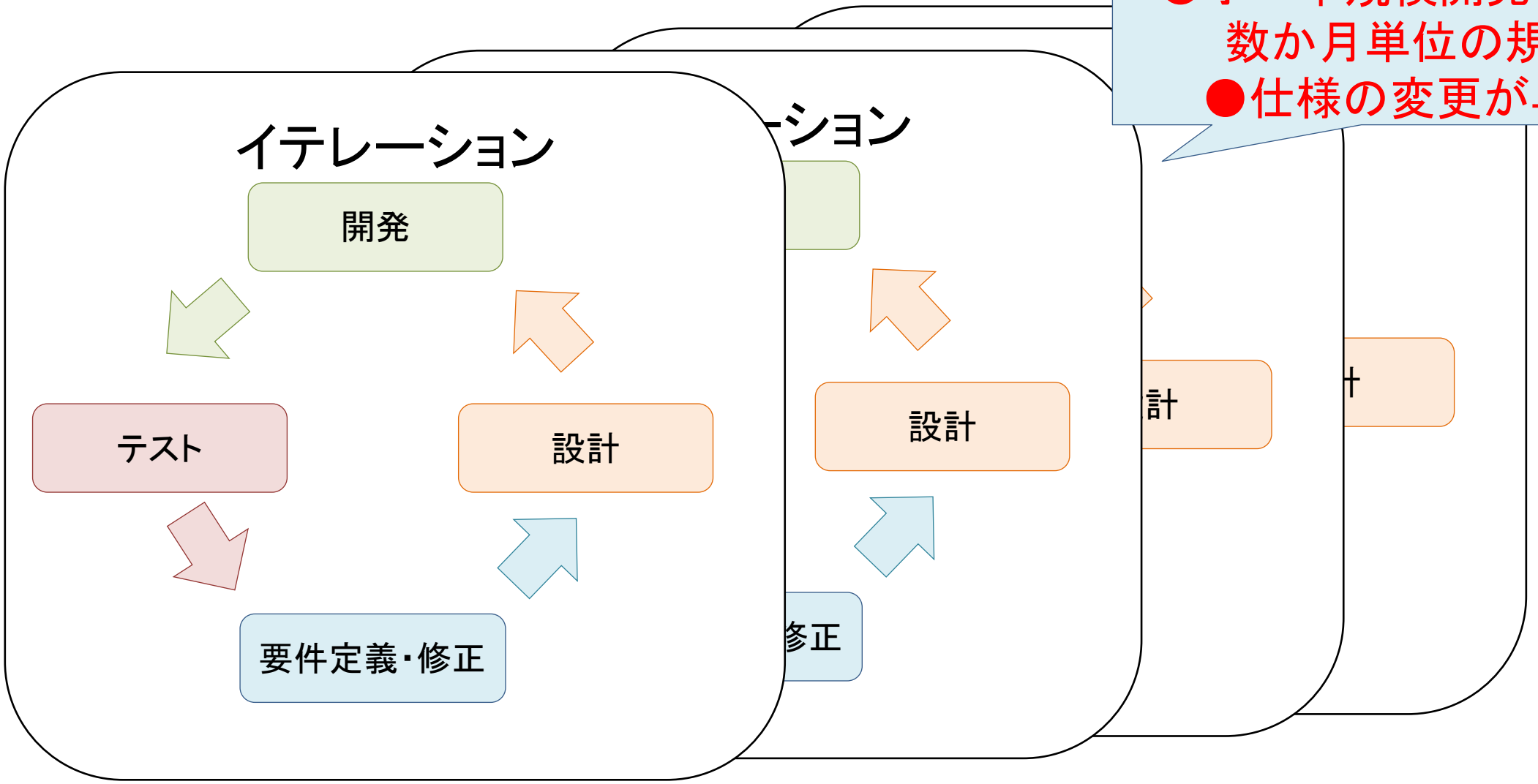


サブシステムに分割後、
ウォータフォールをする。
早期の段階でプログラム
を提供するのは
ユーザの意見を聞くため。

●中規模開発モデル
数か月単位の規模
●複数の希望を並列開発

アジャイル開発モデル

●小～中規模開発モデル
数か月単位の規模
●仕様の変更が早い

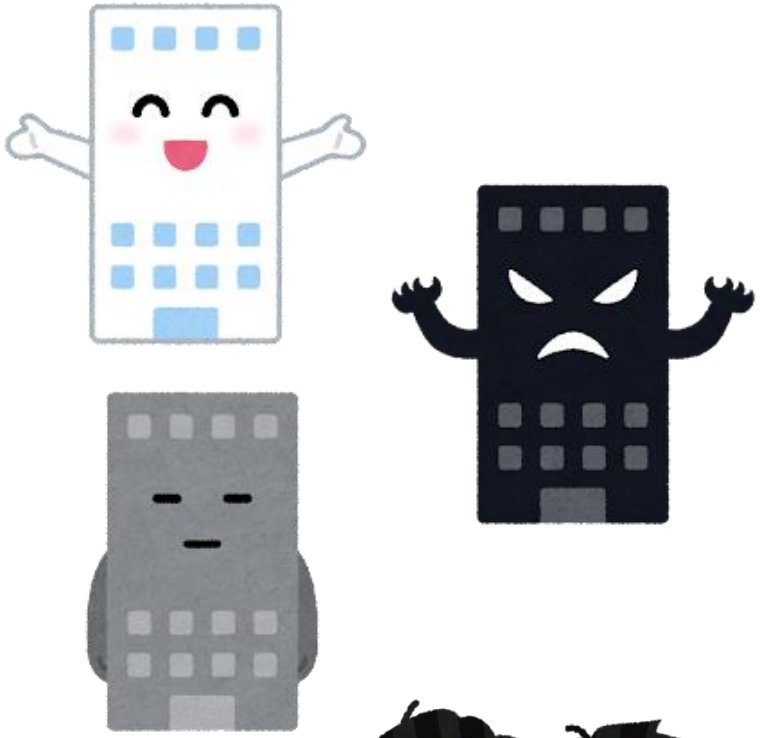


素早い修正と試作を繰り返しながら、開発を進める方法

ペアプログラミング

2人でコーディングを進める方法で、コーディングとレビューを同時にできる
※2人の力量差があると、失敗することもある

の前に



仕事の手順が会社ごとに違うと一緒に仕事をやるときに非常に不便である。

そこで・・・

共通フレーム

システム要件定義
システム方式設計
ソフトウェア要件定義
ソフトウェア方式設計
ソフトウェア詳細設計
ソフトウェア構築
ソフトウェア結合
ソフトウェア適格性確認
ソフトウェア導入
ソフトウェア受入支援
システム結合
システム適格性確認
システム導入
システム受入支援

要求定義：顧客と話し合い、どんなシステムにするかを話し合うこと
システム＝ソフト＋ハードという雰囲気

ファンクションポイント法

画像表示機能	30工程
アラート機能	24工程
ファイル出力	10工程

システムの機能によって作業量を見積もる方法

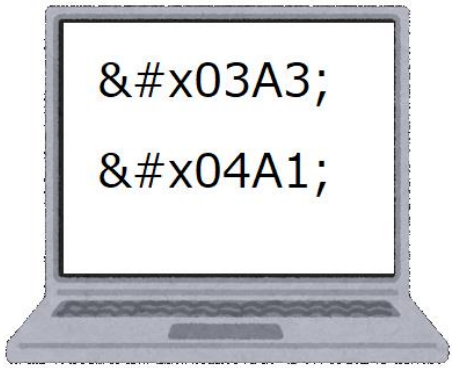
類推見積法

過去の資料を参考にして作業量を見積もる方法

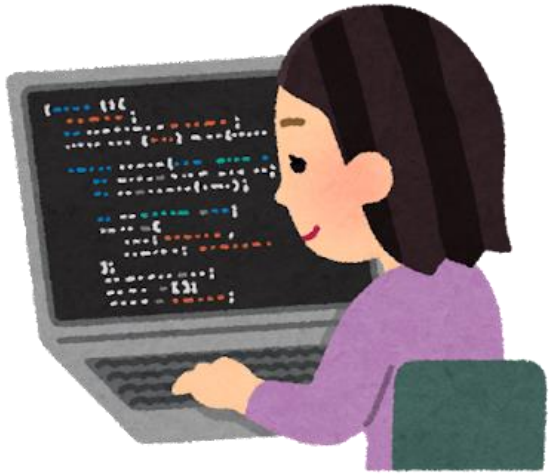
プログラムステップ法

プログラムの量を参考にして作業量を見積もる方法

②高級言語
人が分かるプログラム



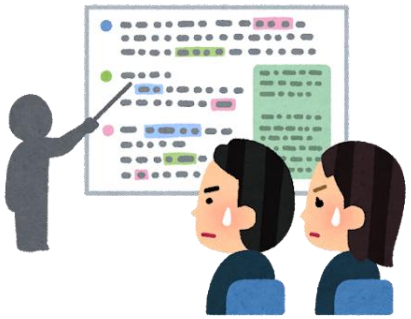
④機械言語
機械が実行する
プログラム



①コーディング
プログラムを書くこと

変換

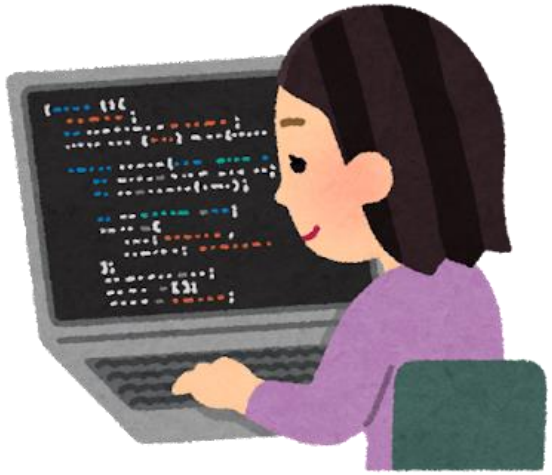
③コンパイラ
高級言語を機械語に変換する仕組み（ふつうはソフトウェア）



コードレビュー
コードを確認すること

デバッグ
プログラムのミスを直すこと

バグが発生



変換

コーディング
プログラムを書くこと

コンパイラ
高級言語を機械語に変換する仕組み（ふつうはソフトウェア）

Twitterからツイートを抜き出すコード

```
CONSUMER_KEY="jXEXOI4MjE44enhbJOO7httXV"
auth.set_access_token(ACCESS_TOKEN,
ACCESS_SECERET)
q = f"ウクライナ exclude:retweets -filter:replies"
tweet_data = []
for tweet in tweepy.Cursor(api.search, q=q,
result_type='recent',locale="ja",lang="ja",include_entities=Fal
se,count=5000).items(5000):


plt.figure(figsize=(15,12))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Twitterからツイートを抜き出すコード

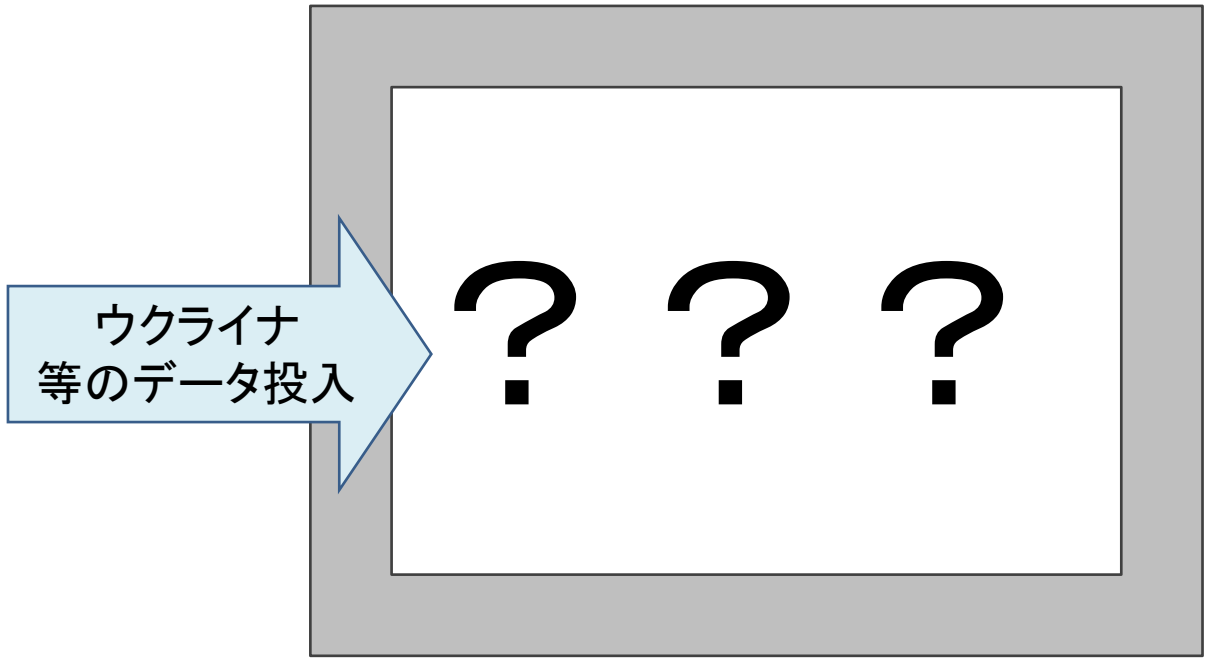
ウクライナを
抜き出す
モジュール

```
CONSUMER_KEY=″jXEXOI4MjE44enhbJ0O7httXV″
auth.set_access_token(ACCESS_TOKEN,
ACCESS_SECERET)
q = f″ウクライナ exclude:retweets -filter:replies″
tweet_data = []
for tweet in tweepy.Cursor(api.search, q=q,
result_type='recent',locale=″ja″,lang=″ja″,include_entities=Fal
se,count=5000).items(5000):
```

ワードクラウド
を表示する
モジュール

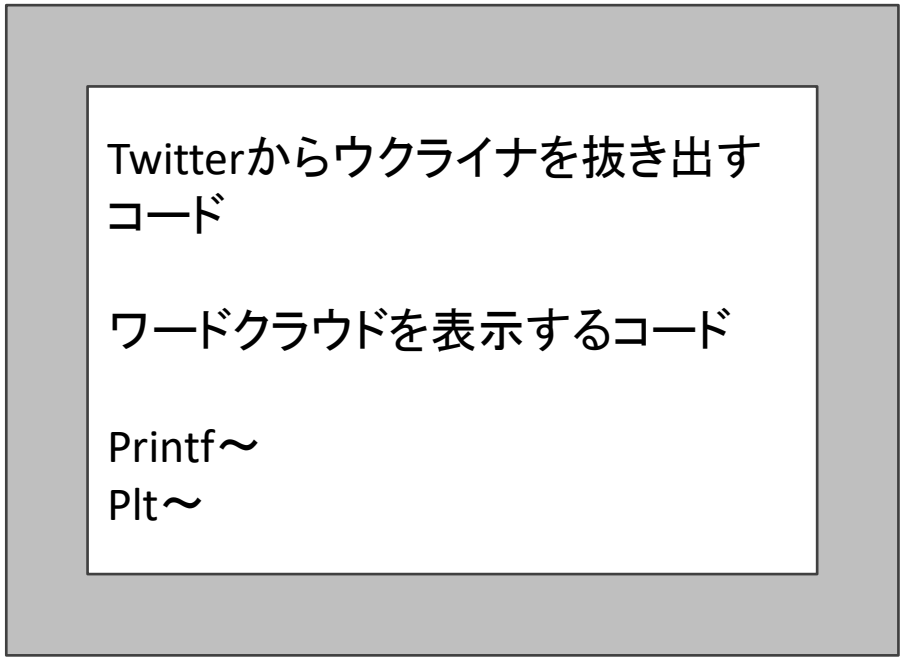
```
plt.figure(figsize=(15,12))
plt.imshow(wordcloud)
plt.axis(″off″)
plt.show()
```

テスト



ブラックボックス

中のプログラムは分からないので、様々な値を入力するテスト

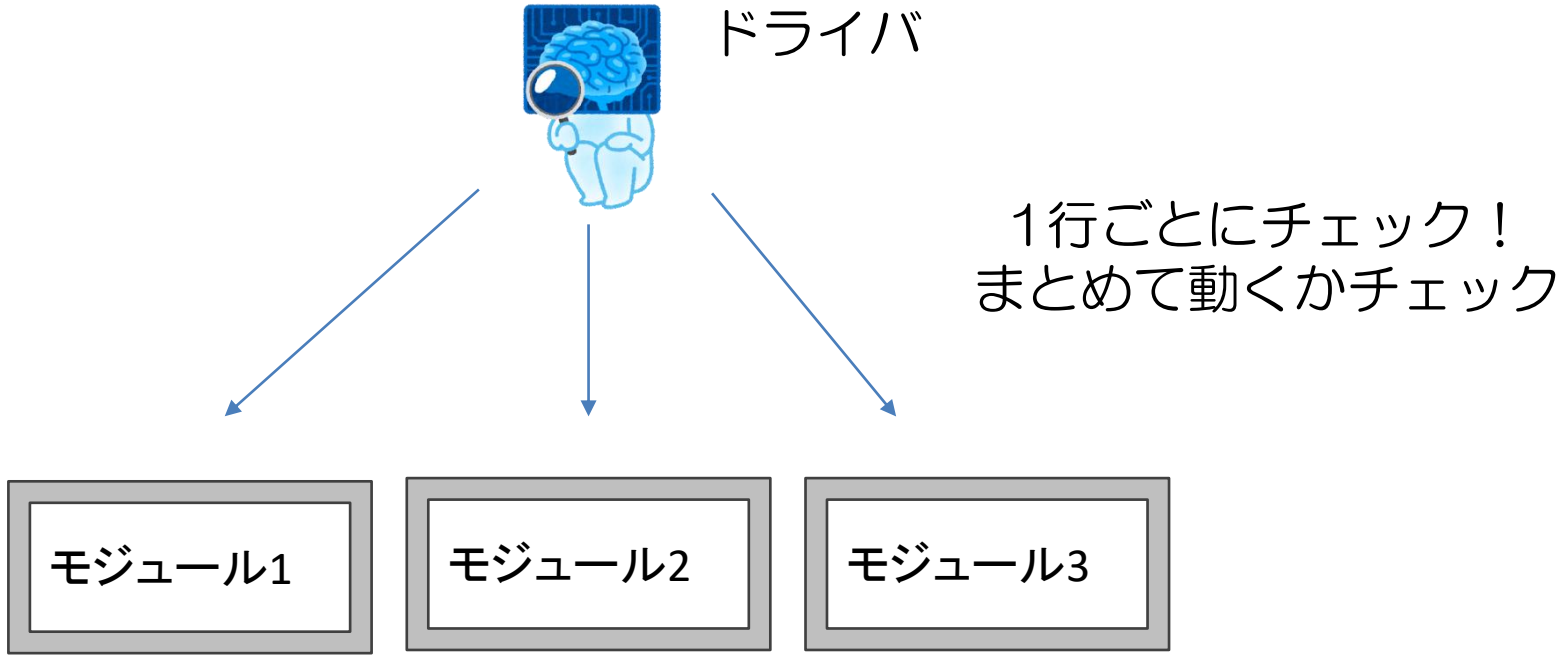


ホワイトボックス

中のプログラムを１つ１つ確認して、要求を満たすか確認するテスト

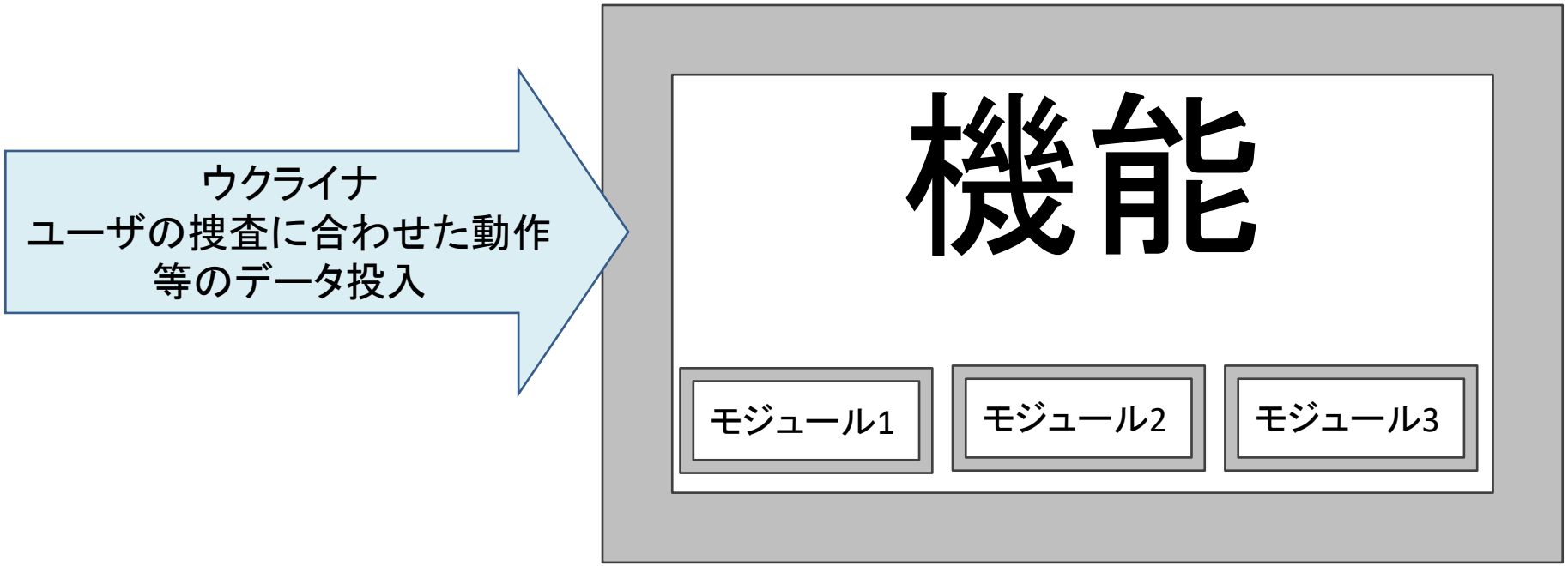
単体テスト

モジュールを1個1個チェックする
ホワイトボックステストを使うことが多い。



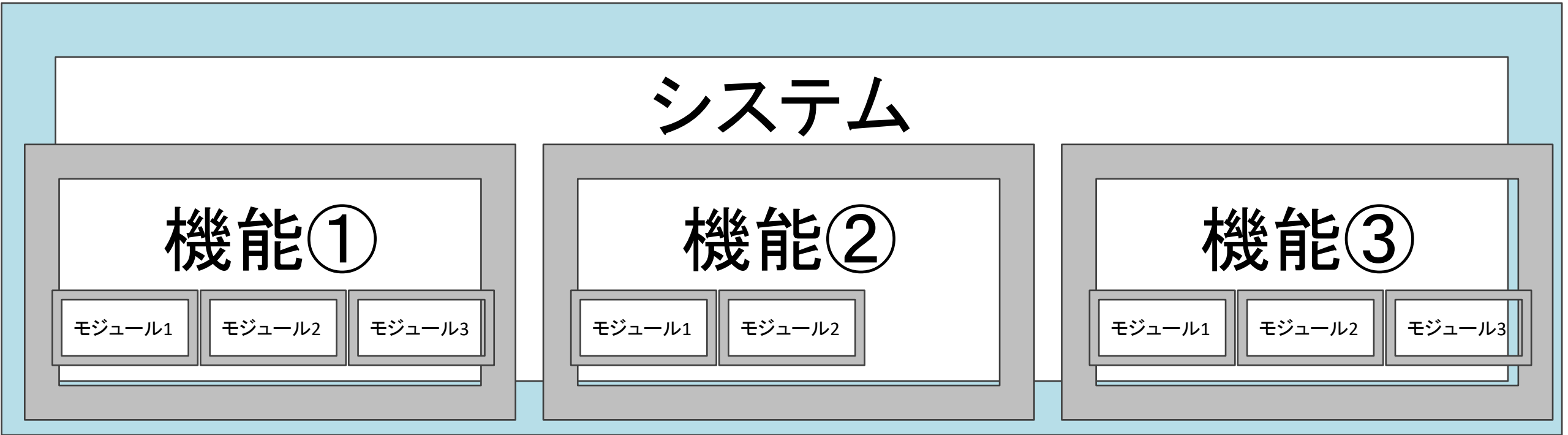
結合テスト

モジュールを合体させるテスト
計画が遅延した場合、スタブを用いる
ブラックボックステストを使うことが多い



システムテスト

どんどん合体させて、システムを構築
（**システム結合テスト**）
そのシステムが起動して、動作を終了するまでを
行うテスト（**システムテスト**）



運用テスト

開発者が責任もって動作確認をするテスト
現場に近い環境でテストを行う

受け入れテスト

ユーザと製作者が動作確認をするテスト
実際の現場に互いが来て、テストを行う