ソフトウェア工学

吉田 則裕

立命館大学 情報理工学部

2025年度 春学期

X: @NorihiroYoshida

第3回: プロジェクト管理（教科書第3章）

- プロジェクト管理における作業プロセス
  - PMBOK
- 開発工数の見積もり技法
  - 標準タスク法
  - COCOMO, COCOMO II
  - ファンクションポイント法(FP法)
- ソフトウェアの品質管理
  - 品質特性
  - メトリクス，GQM
- プロセス成熟度
  - CMM, CMMI

考えて欲しいこと（講義のポイント）

- プロジェクトでは何を管理しなければいけないか？
- 開発規模の見積もり技法にはどのようなものがあるか？
- ソフトウェアの品質とは何か？
- ソフトウェアの品質はどのように計測できるか？
- ある組織の開発プロセスをレベルの高低をどのように評価するか？

管理の多義性

英語圏でも，一貫した使い分けがなされていない．

- Administration
  - 組織全体にとって重要な方針や目標を設定し，それを守らせる．
- Management
  - Administrationによって決められた方針や目標を実行に移すための活動
  - ほっとくと達成できない方針や目標を「なんとかやりくり」すること
  - リスクと意思決定を伴う
  - 多くの場合，人を対象に含む．
  - Managerは多くの場合に人的資源のやりくりをする．
    - 例：project management
- Control
  - 決められた基準に沿って，安定した状態に統制すること
  - あまり人間を対象としない
    - 例：quality control, passport control

プロジェクト管理は何を管理するのか？

QCD(+S)

- 品質(Quality)，コスト(Cost)，納期(Delivery)
- スコープ(Scope)

プロジェクト管理とは

- ソフトウェア開発において，「誰が」「どの段階で」「何をすればよいのか」を計画する作業
  - 要求を満たすソフトウェアを，決められた予算と期間で納品することを目的
- 計画どおりにプロジェクトが進められているかを確認し，その結果に応じた対策作業
- プロジェクト管理
  - QCD(+S)を中心
    - 品質(Quality)，コスト(Cost)，納期(Delivery)
    - スコープ(Scope)
- PMBOK(project management body of knowledge)
  - プロジェクト管理に必要な知識を9つの視点で体系化
    - 統合管理，スコープ管理，時間管理，コスト管理，品質管理，人材管理，コミュニケーション管理，リスク管理，調達管理

BOK = Body of Knowledge = 基礎知識体系

PMBOKによるプロジェクトの作業プロセス

- 立ち上げプロセス
  - プロジェクトまたはそのフェーズを定義し，認可
- 計画プロセス
  - プロジェクトの目標を定義・洗練
  - 活動計画の策定
  - 資源の調達
- 遂行プロセス
  - プロジェクト計画を実施
- 制御プロセス
  - プロジェクト計画との差異を認識するために，実績報告や進捗測定を実施
- 終結プロセス
  - 契約の完了手続きを行い，プロジェクトを終了

開発計画で明確にすべき項目

- 開発の目的
  - 何のためにこの開発が行われるのか
- 開発の目標
  - システム利用者の要望や業務運営上の方針など
- 開発対象業務および運用方針
  - 開発対象の範囲と機能など
  - 業務上の制約や利用者に関する前提条件など
- 開発システムの基本構成
  - 開発するソフトウェアが実際に稼働する環境など
- 開発工数
  - 開発に要する工数と開発コストとそれらの見積もり
- 開発スケジュール
  - 開発工数に応じた開発期間の決定
    - 作業明細構造(WBS: work breakdown structure)やガントチャートの活用

開発計画で明確にすべき項目 (cont'd)

- 開発体制
  - 開発工数とスケジュールに応じた組織やチーム
  - 必要要員の割り当て計画
- 開発環境や開発方法
  - 開発支援ツール
  - 開発方法論，コーディング規約など
- 成果物の管理方法
  - 各工程で作成された成果物を管理(構成管理)する方法
- リスク管理の方法(risk management)
  - 開発を進めていくうえで発生が予想されるリスクとその対処方法
    - リスク衝撃(risk impact): リスクによる損失
    - リスク確率(risk probability): リスクが起こる可能性
    - リスク制御(risk control): リスクの影響を最小化，回避する行動

工数見積に失敗するとどうなるか？

→プロジェクトが炎上する.

- 工数見積はとっても大切
- 工数に余裕があれば，トラブルが起きても対策できる
- 工数見積の研究は数多くあり，企業でもさまざまな取り組みが行われている.

開発工数の見積もり技法

- 標準タスク法
  - ソフトウェア開発に必要とされる標準的な作業(標準タスク)ごとに開発工数をあらかじめ設定
  - 実際の開発に現れる作業を予測し，それらに応じて工数を積み上げることで全体の開発工数を算出
- COCOMO(constructive cost model) [Boehm]
  - 開発ソフトウェアの規模(予測規模)から開発工数を算出
    - 開発規模と開発工数の統計的モデルを利用
- ファンクションポイント法(FP法) [Albrecht]
  - ソースコードの行数の代わりに，入力や出力などの機能の数(function point)から規模を算出
  - 開発工数への変換モデルは未提示
- COCOMO II
  - オブジェクトポイントやファンクションポイントから得られるソフトウェアの規模から開発工数を算出

標準タスク法による開発工数の算出

- 各標準タスクに対して，規模と複雑度に応じた件数を数え上げ

(a) 標準タスクAの作業日数

規模＼複雑度	単純	普通	複雑
小	1	2	3
中	1.5	3	5
大	2	4	7

(b) プロジェクトXにおける標準タスクAの件数

規模＼複雑度	単純	普通	複雑
小	10	5	0
中	10	30	5
大	0	10	10

規模＼複雑度	単純	普通	複雑
小	1×10	2×5	3×0
中	1.5×10	3×30	5×5
大	2×0	4×10	7×10

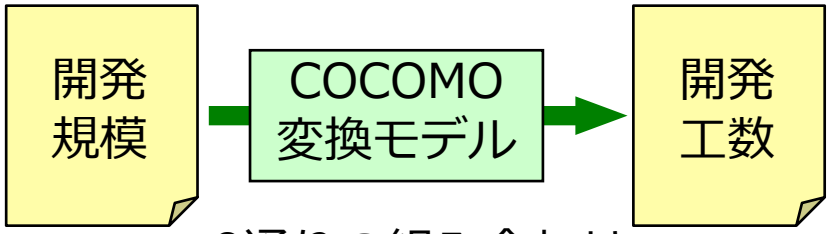
(c) プロジェクトXにおける標準タスクAの工数

10+10+0+15+90+25+0+40+70
= 260 (標準タスクAの総工数)

プロジェクトXの総工数
= 標準タスクAの総工数
+ 標準タスクBの総工数
+ 標準タスクCの総工数
+ ….

COCOMO

- COCOMO変換モデル
  - ソースコードの行数から見積もった開発規模を開発工数に変換
- 見積もりの実施時期に基づく変換モデル
  - 基本COCOMO (プロジェクトの計画段階で利用)
    - 開発規模のみから算出
  - 中間COCOMO (要求定義が完了した後で使用)
    - 開発するソフトウェアの特徴やメンバの経験や能力で調整して算出
  - 詳細COCOMO (設計終了後に利用)
    - モジュール構成などを考慮し，開発規模を調整して算出
- 開発の規模に基づく変換モデル
  - オーガニックモード (単純・小規模)
  - 半組込みモード (一般的)
  - 組込みモード(複雑・大規模)



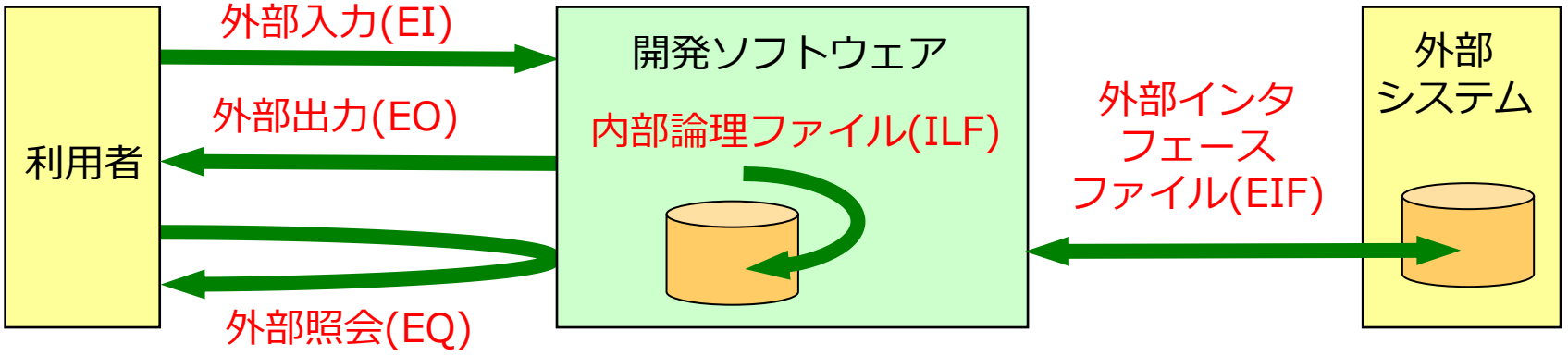
9通りの組み合わせ

COCOMO II

- COCOMOではソースコードの規模から工数を算出
  - 分析・設計工程でソースコードの規模を予測することは困難
- COCOMO IIではオブジェクト数や機能数に基づき算出
- 3つの変換モデル
  - アプリケーション組み立てモデル
    - オブジェクトポイント(オブジェクトの登場数に重みをつけた数値)で規模を算出
    - プロトタイプ開発において適用
  - 初期設計モデル
    - ファンクションポイント法に基づく機能数で規模を算出
    - システム構造を決定する前に適用
  - ポストアーキテクチャモデル
    - ファンクションポイント法に基づく機能数やソースコードの行数で規模を算出
    - システム構造を決定した後に適用

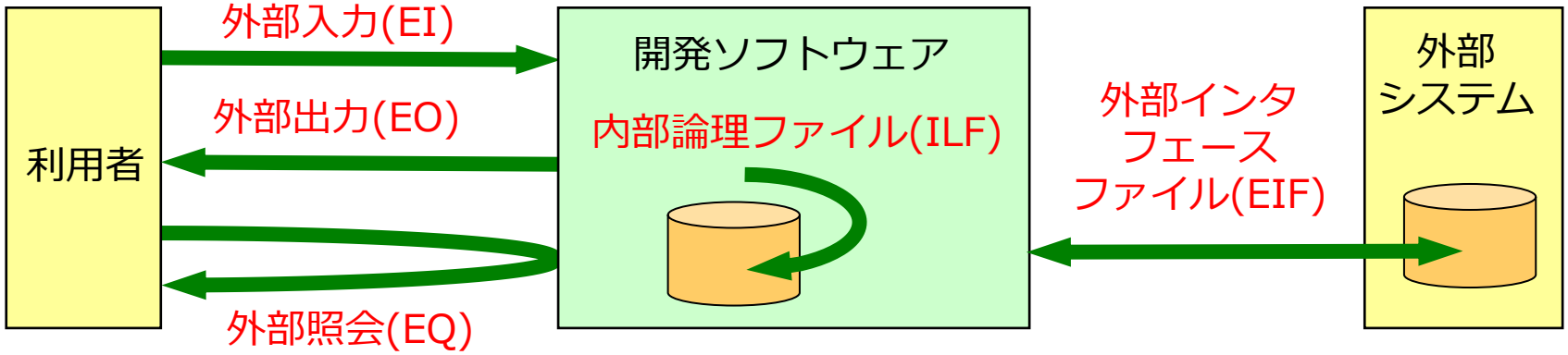
ファンクションポイント法(FP法)

- ソフトウェア内部の処理を5つの機能に分類して規模を算出



EI: 外部から開発ソフトウェアへの入力(内部論理ファイルの更新あり)
EO: 開発ソフトウェアから外部(利用者など)への出力
EQ: 外部から開発ソフトウェアへの照会(内部論理ファイルの更新なし)
ILF: 開発ソフトウェア内部に存在するファイルの規模(論理レコードの数)
EIF: 他のシステムで管理され，開発ソフトウェアが参照するファイルの規模(論理レコードの数)

FP値の算出 (1/3)



EIの複雑度

関連ファイル数	データ項目数		
	1～4	5～15	16～
0～1	単純	単純	普通
2	単純	普通	複雑
3～	普通	複雑	複雑

EO, EQの複雑度

関連ファイル数	データ項目数		
	1～5	6～19	20～
0～1	単純	単純	普通
2	単純	普通	複雑
3～	普通	複雑	複雑

ILF, EIFの複雑度

レコード数	データ項目数		
	1～19	20～59	60～
0～1	単純	単純	普通
2	単純	普通	複雑
3～	普通	複雑	複雑

FP値の算出 (2/3)

- 5つの機能に対して，3つの複雑度ごとに機能数を数え上げ
- 複雑度別の機能数に重み付け係数を掛け，合計値を算出

(a) 複雑度別の機能数

規模＼複雑度	単純	普通	複雑
EI	10	9	15
EO	14	12	14
EQ	2	4	7
ILF	3	3	6
EIF	4	5	5

(b) 重み付け係数

規模＼複雑度	単純	普通	複雑
EI	×3	×4	×6
EO	×4	×5	×7
EQ	×3	×4	×6
ILF	×7	×10	×15
EIF	×5	×7	×10

規模＼複雑度	単純	普通	複雑
EI	10×3	9×4	15×6
EO	14×4	12×5	14×7
EQ	2×3	4×4	7×6
ILF	3×7	3×10	6×15
EIF	4×5	5×7	5×10

(c) ファンクションポイント

30+36+90 + …
= 680 (未調整FP)

FP値の算出 (3/3)

- システムの特性に応じて得点を割り当て，FP値を調整

システム特性	得点
1. データ通信	0
2. 分散処理	2
3. パフォーマンス	4
4. 高負荷環境	4
5. トランザクション量	2
6. オンラインデータ入力	2
7. エンドユーザの作業効率	0
8. マスターデータベースのオンライン更新	1
9. 内部処理の複雑さ	1
10. 再利用を考慮した設計	1
11. 導入の容易性	3
12. 運用の容易性	3
13. 複数サイトでの使用	2
14. 変更の容易性	3

0: まったく関係ない
1: ほとんど影響を受けない
2: 適度に影響を受ける
3: 平均的な影響を受ける
4: 大きな影響を受ける
5: 非常に大きな影響を受ける

合計 = 28

調整用係数C
= 0.65 + (0.01 × 28)
= 0.93

FP
= 0.93 × 680
= 632.4

FP値の実際

- 実際に企業で使われている
- FP値を計測できる人材を育成・確保しなければならない

品質とは何か？

- 1枚のレーズンの数はどれも概ね同じ
- パンの厚さは，概ね同じ（誤差0.01mm以内）
- JIS Q 9000:2015
  - 「対象に本来備わっている**特性の集まり**が，要求事項を満たす程度」

ソフトウェアの品質特性(ISO 9126)

ソフトウェアの品質には，複数の品質特性がある．

- 機能性(functionality): 必要な機能が実装されているか
- 信頼性(reliability): 機能が正常に動作し続けるか
- 使用性(usability): 利用者にとって使いやすいか
- 効率性(efficiency): 目的達成のために使用する資源は適切か
- 保守性(maintainability): 改訂作業に必要な労力は少ないか
- 移植性(portability): 他の環境へ移しやすいか

一つの品質特性に複数の副特性がある

- 機能性(functionality): 必要な機能が実装されているか
  合目的性，正確性，相互運用性，セキュリティ，機能性，標準適合性
- 信頼性(reliability): 機能が正常に動作し続けるか
  成熟性，障害許容性，回復性，信頼性，標準適合性
- 使用性(usability): 利用者にとって使いやすいか
  理解性，習得性，操作性，魅力性，使用性，標準適合性
- 効率性(efficiency): 目的達成のために使用する資源は適切か
  時間的効率性，資源効率性，効率性，標準適合性
- 保守性(maintainability): 改訂作業に必要な労力は少ないか
  解析性，変更性，安定性，試験性，保守性，標準適合性
- 移植性(portability): 他の環境へ移しやすいか
  順応性，設置性，共存性，置換性，移植性，標準適合性

品質管理(Quality Management)

- ソフトウェアの品質を向上させる作業
  - プロジェクト管理とは独立の組織で実施

  - 品質保証(quality assurance)
    - 高品質ソフトウェアの提供するための組織的な手続きや基準のフレームワークを構築

  - 品質計画(quality planning)
    - 適切な手続きや基準の選択と開発プロジェクトへの適合

  - 品質管理(quality control)
    - 計画した手続きや基準に開発チームが従うことを保証するためプロセスの定義や執行

ソフトウェアメトリクス

- ソフトウェアの品質を管理するためには，ソフトウェア開発におけるプロダクトやプロセスの評価が必須
  - 開発の実態を定量的に評価する手段(数値，尺度，属性など)
  - 開発標準との比較や改善に利用

プロダクト/プロセスメトリクス

- プロダクトメトリクス(product metrics)
  - ソースコードの規模
    - LOC(lines of code): 行数
    - ステップ数: プログラムのだけを数えたもの
    - Halsteadの尺度 [Halstead]: プログラム中の演算子と非演算子の種類数や出現回数から計測
  - 複雑さ
    - サイクロマティック数(cyclomatic number) [McCabe]
    - CKメトリクス [Chidamber & Kemerer]

- プロセスメトリクス(process metrics)
  - 作業効率を計測
    - 作業に費やした時間や資源の量
    - 特定のプロセスの実行中に発生したイベントの数(コードインスペクションで発見された誤りの数など)

尺度の種類（名義尺度・順序尺度）

- 名義尺度
  - 対象物に数値を名前をとして割り当て
  - 値の大きさに意味はない
  - メトリクス例：使用言語
    - C++:1, Java:2, Python:3, その他:4

- 順序尺度
  - 対象物間で順序関係が成立するように数値を割り当て
  - メトリクス例：システム障害のレベル
    - 致命的:3, 重大:2, 軽微:1

尺度の種類（間隔尺度・比例尺度）

- 間隔尺度
  - 順序だけでなく，値の差の意味がある.
  - メトリクスではほとんどない.
    - 例：温度（摂氏や華氏）
      - 温度が上昇したとき，その差には意味がある
      - 温度が上昇したとき，上昇率には意味がない

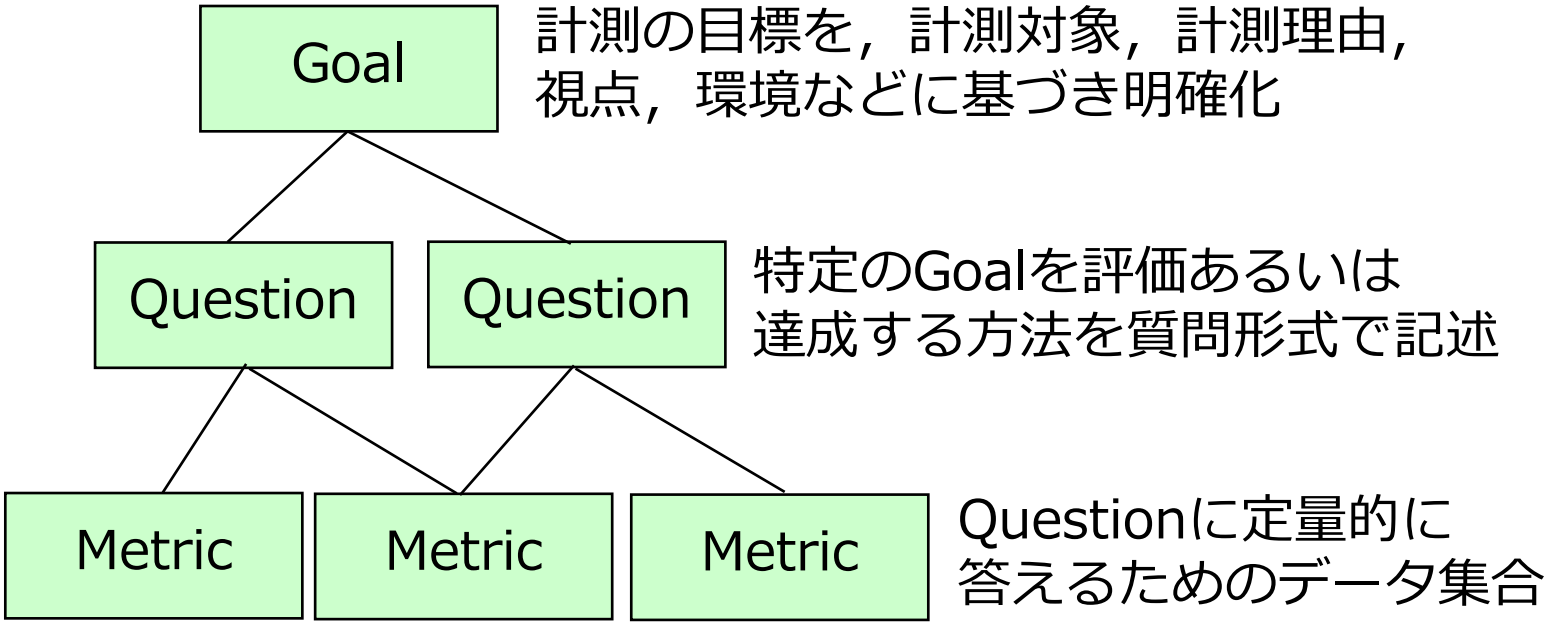
- 比例尺度
  - 値の比に意味がある
  - メトリクス例：コード行数

企業におけるメトリクスの使用例

- ソースコードの品質基準を決めておき，満たさない場合は出荷させない
  - 基準は，工場や製品種別ごとに決められる
- 開発チームとは別の部門が検査を行う
- 自社製品の付加価値を向上させるために，品質保証に熱心な企業が多い

GQM(Goal-Question-Metric)

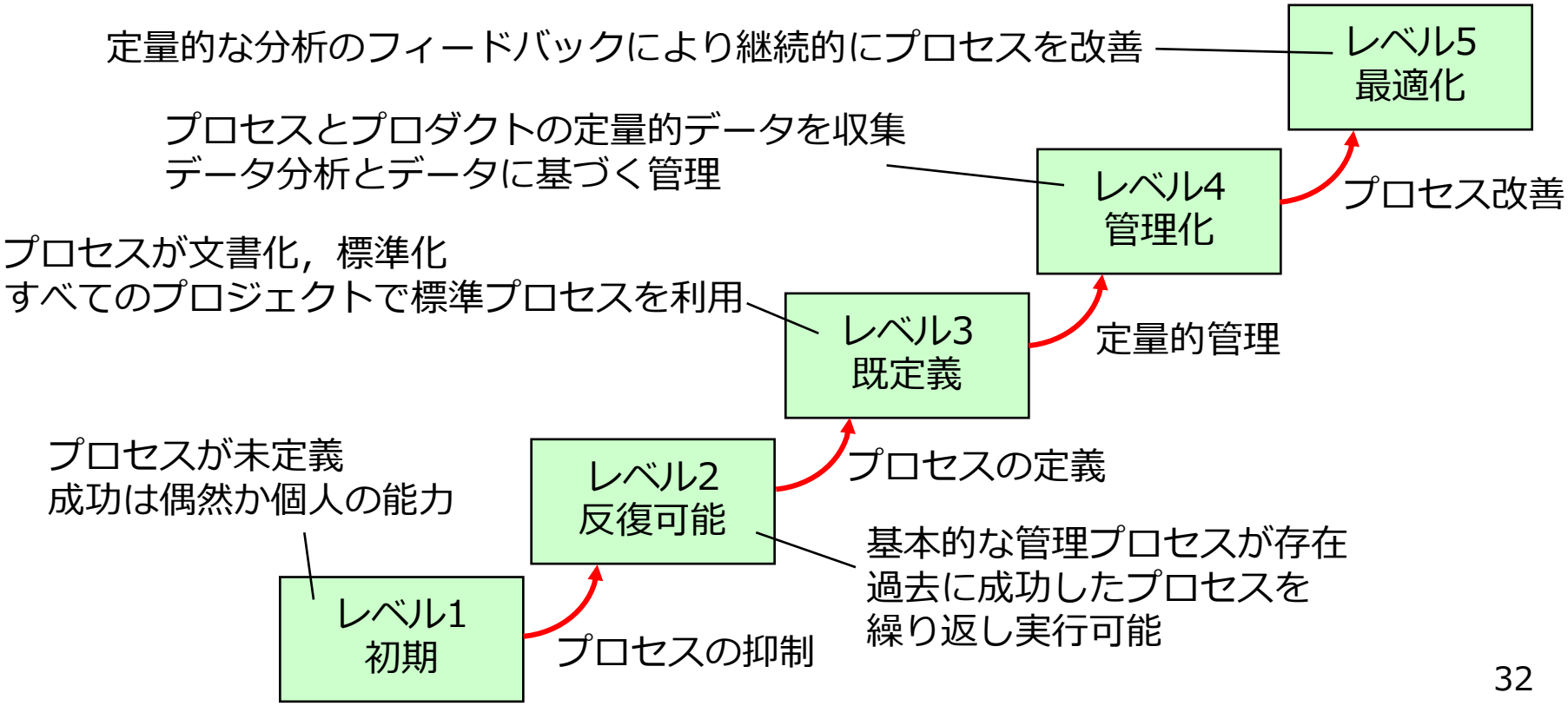
- 計測は目的をもって実行されるべき
- 計測目標とメトリクスの対応関係を明確化
  - Goalにはテンプレートがあらかじめ用意
  - コンテキストが重要



余談：授業評価でも同じような概念がある

プロセス成熟度

- プロダクトの品質を高めるためにプロセスの品質を改善
  - CMM(capability maturity model) [SEI]
    - 現在はCMMI(CMMI Integration)
  - SW-CMM: CMMIのうちソフトウェアに関係するもの
    - 成熟のレベルを5段階で規定



プロセスの外部評価は難しい

外部からの評価はどうしても，書類審査がメインになりがち・・・

まとめ

- プロジェクト管理とは何か？
- 開発工数の見積もり技法
  - 標準タスク法
  - COCOMO, COCOMO II
  - ファンクションポイント法(FP法)
- ソフトウェアの品質管理
  - 品質特性
  - メトリクス，GQM
- プロセス成熟度
  - CMM, CMMI