第4回SEA上海　ソフトウェアフォーラム

uVALUE

プロジェクトレベルの改善と組織レベルの改善

2009/12/11

HITACHI
Inspire the Next

株式会社 日立製作所　福地　豊

uVALUE

Contents

HITACHI
Inspire the Next

1 企業の目的

（1） 利益をあげる

・原価を低減する C

・売価を増加する

⇒ 受注前に高い精度で原価を見積もる

（2） 企業として継続しつづける

・品質の高い製品を提供する Q

・納期通りに納入する D

⇒ 高い開発技術がある

⇒ 開発のプロセスをコントロールできる

（3） 社会に貢献する

・良い製品を提供する

・情報発信する

・標準化に寄与する　等

⇒ 事例を収集・整理できる

1 企業の目的

ソフトウェア開発を行う組織の方針

例えば

- CMMI レベル5と認定される
- 基準を整備し、基準通りに開発を行う
- 自社開発を行い技術を蓄積する
- 人材を育成する　等

1 企業の目的

・CMMI レベル5と認定される

レベル5と認定されることだけを目標としても意味が無い

最初からレベル5と認定されることはあり得ない

レベル5に向けて段階的な計画を行う必要がある

プロセス改善の仕組みを組織的に組み込む

どのように改善の仕組みを作るか？

1 企業の目的

・基準を整備し、基準通りに開発を行う

(a) 一般公開された基準・数値情報を利用
(b) 別の企業の基準・数値情報を利用

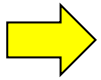
自組織の特質と必ずしも一致するとは限らない
（他の企業と同じことをしても生き残れない）

初期のスタートアップとして利用し、自組織の基準・数値情報に発展させる。

2 プロジェクト

プロジェクトを実施できる条件

・技術がある
・人材が揃っている
・プロジェクトの管理ができる
・開発機材が準備できる



とりあえず
開発ができる

〇プロジェクトが成功するかどうかは個人の
力量により決まる。
〇最後まで、成功するかどうか予測できない。

CMMIでは、管理ができる事をレベル2と定義
ISO/IEC 15504-5 のExampleでは開発ができる事をレベル2としている
（技術先行）

3 プロジェクトでのデータ収集

- 「プロジェクトの管理ができる」にはどうすれば良いか

> 開発の状況を表現できる指標により、
> プロジェクトを制御する

指標の考え方

QCDを中心に指標を収集する

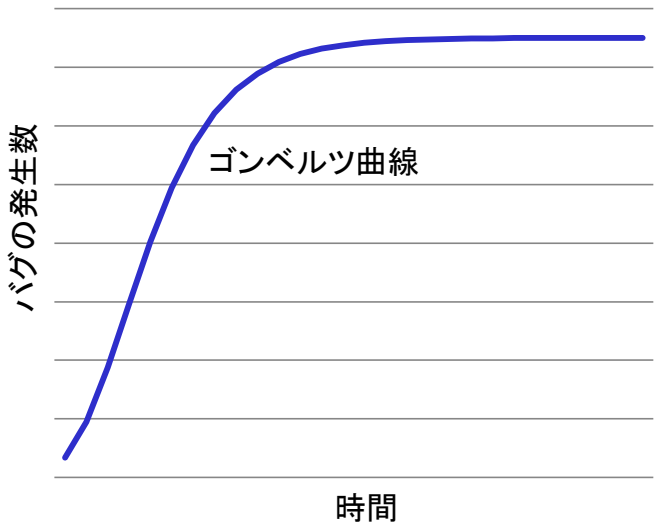
Q : Quality	チェックリストの件数、バグの件数、バグの種類、バグの原因
C : Cost	規模、工数
D : Delivery	進捗度、バグの発生状況、バグ解決状況

3 プロジェクトでのデータ収集

Q : Quality

チェックリストの件数、バグの件数、
バグの種類、バグの原因

コントロールするには、
予測値と実績値が必要



品質評価

工程の完了時点での評価

予測値よりも多い／少ない

多くても少なくても品質に問題がある可能性がある

品質管理

時系列なテスト状況の管理

3 プロジェクトでのデータ収集

Q : Quality

チェックリストの件数、バグの件数、
バグの種類、バグの原因 ⇒ 規模当り

指標	指標を決める要素	測定方法の考慮点
チェックリストの件数	工程、規模、チェック項目種類	チェック項目の数え方
バグの件数	工程、規模	バグの数え方 Ex) 1カ所で発見されたバグが複数のプログラムに影響がある場合、1件にするか複数件にするか？
バグの種類	工程	複数の種類に関係ある場合の数え方
バグの原因	工程、本来発見すべき工程	複数の原因がある場合の数え方

品質に関する規格は ISO/IEC 9126 , ISO/IEC 25000シリーズ(SQuaRE)が参考にできる

3 プロジェクトでのデータ収集

Q : Quality

チェックリストの件数、バグの件数、
バグの種類、バグの原因 ⇒ 規模当り

指標を決める要素	考慮点	備考
工程	・プロジェクトでの工程定義 ・標準の工程の有無 ・標準のプロセスの有無 ・工程の入出力情報の定義 ・取得する指標	開発プロセスは、ISO/IEC 12207が参考にできる。

3 プロジェクトでのデータ収集

C : Cost

規模、工数

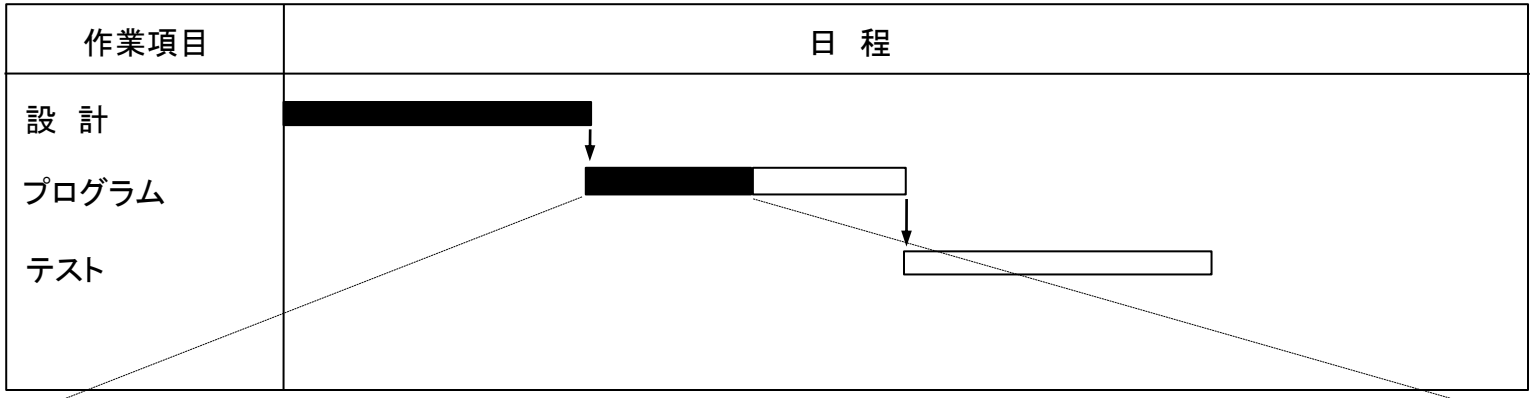
コントロールするには、
予測値と実績値が必要

指標	考慮点	備考
規模	・FP (Function Point) ・ステップ数 コメントの有無 単純な行数／論理的な行数 ファイル単位／メソッド単位 includeの数え方 定義の数え方 （XML,パラメータ）	<論理的な行数> SQLの例 SELECT a,b,c form table1 where id = “x” →複数行に渡る場合でも1行として計測する場合
工数	・人数 ・働いた時間 ・プロジェクトで働いた時間 ・間接的に関わる人 ・管理者	<収集方法> 仕事の分類毎に時間を申告する。 自動的に収集する。

3 プロジェクトでのデータ収集

D : Delivery

進捗度、バグの発生状況、バグ解決状況



	規模 (Kstep)	チェックリスト消化予定	チェックリスト消化数	進捗度 (%)	進み/遅れ日数	バグ件数予定	バグ件数
プログラムA	500	50	30	60	0	30	0
プログラムB	600	60	20	33	0	36	50
プログラムC	550	55	60	109	-2	33	70
プログラムD	400	40	30	75	-1	24	20
プログラム全体	2,050	205	140	68	-0.75	123	140

4 プロジェクトでのプロセス改善

プロジェクトは計画通りに実施されているかをチェックしながら進むので、容易には変更ができない。

	規模 (Kstep)	チェックリスト消化予定	チェックリスト消化数	進捗度 (%)	進み/遅れ日数	バグ件数予定	バグ件数
プログラムA	500	50	30	60	0	30	0
プログラムB	600	60	20	33	0	36	50
プログラムC	550	55	60	109	-2	33	70
プログラムD	400	40	30	75	-1	24	20
プログラム全体	2,050	205	140	68	-0.75	123	140

規模の取得に手間がかかるので、自動計測する様に工夫

チェックリストの妥当性を確認するために、レビュー観点を決め、レビューを行う

進み遅れ日数の計算が実態を表していないので、計算方法を見直しする

バグの発生原因を分析するため、バグの原因別件数を取得する

4 プロジェクトでのプロセス改善

プロジェクトは計画通りに実施されているかをチェックしながら進むので、容易には変更ができない。

	規模 (Kstep)	チェックリスト消化予定	チェックリスト消化数	進捗度 (%)	進み/遅れ日数	バグ件数予定	バグ件数
プログラムA	500	50	30	60	0	30	0
プログラムB	600	60	20	33	0	36	50
プログラムC	550	55	60	109	-2	33	70
プログラムD	400	40	30	75	-1	24	20
プログラム全体	2,050	205	140	68	-0.75	123	140

規模の取得に手間がかかるので、自動計測する様に工夫

チェックリストの妥当性を確認するために、レビュー観点を決め、レビューを行う

進み遅れ日数の計算が実態を表していないので、計算方法を見直しする

バグの発生原因を分析するため、バグの原因別件数を取得する

4 プロジェクトでのプロセス改善

プロジェクトは計画通りに実施されているかを、チェックしながら進むので、容易には変更ができない。

次のプロジェクトで、現在進行中/完了のプロジェクトの改善可能な部分を反映する。

データ収集方法　:プロセスの見直し、フォーマットの見直し

状況整理方法　　:フォーマットの見直し、分析方法の見直し

データの利用　　:実績データの整理、見積りに利用

5 組織レベルの改善

プロジェクトでのプロセス改善は、属人的になっており、そのプロジェクトを実施した人にしか分からない。

データ収集方法　：プロセスの見直し、フォーマットの見直し

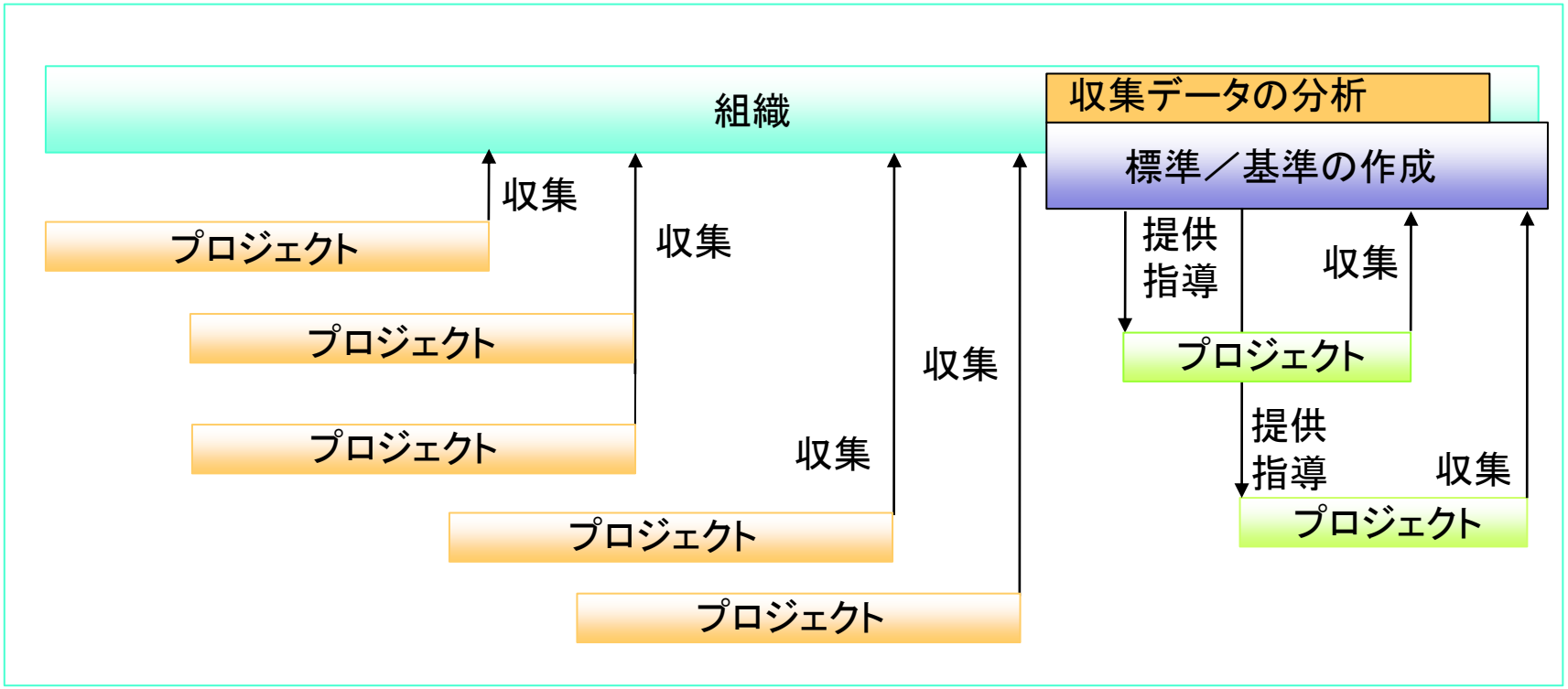
状況整理方法　　：フォーマットの見直し、分析方法の見直し

データの利用　　：実績データの整理、見積りに利用

プロジェクト毎に異なった
プロセス、フォーマット、分析方法、収集データ、
見積りのベースとなるプロフィール情報
（顧客、業務、インフラ、環境、外部委託、標準/基準の整備状況等）

5 組織レベルの改善

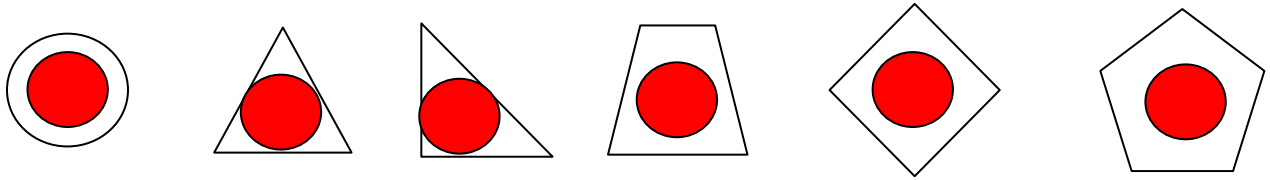
組織レベルの改善のサイクル



プロジェクト毎にバラバラに収集しているデータや、作成している標準を集めても、組織としての標準や基準を作成するのは難しい。

5 組織レベルの改善

収集データのばらつき

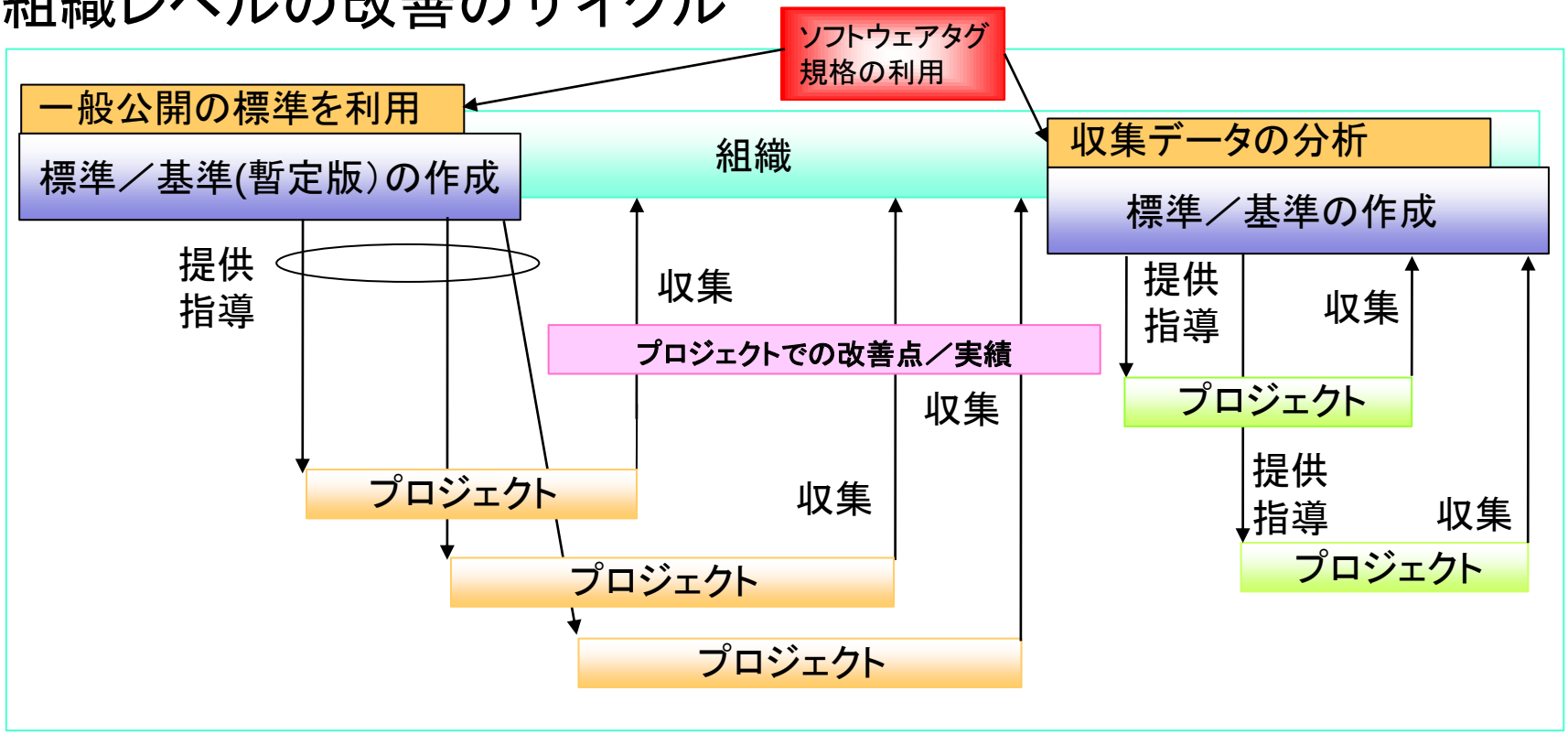


プロジェクト	規模の測定	備考
プロジェクトA	ファイル数、ステップ数、定義文の行数	
プロジェクトB	ファイル数、ステップ数	
プロジェクトC	クラス数、メソッド数、ファイル数、ステップ数、自動生成ステップ数、改造ステップ数	
プロジェクトD	FP数	完了時にもFP計測

最大公約数である、ステップ数を収集対象とする
プロジェクトDの見積り精度により、見積りにはFPを利用することを制度化する

5 組織レベルの改善

組織レベルの改善のサイクル

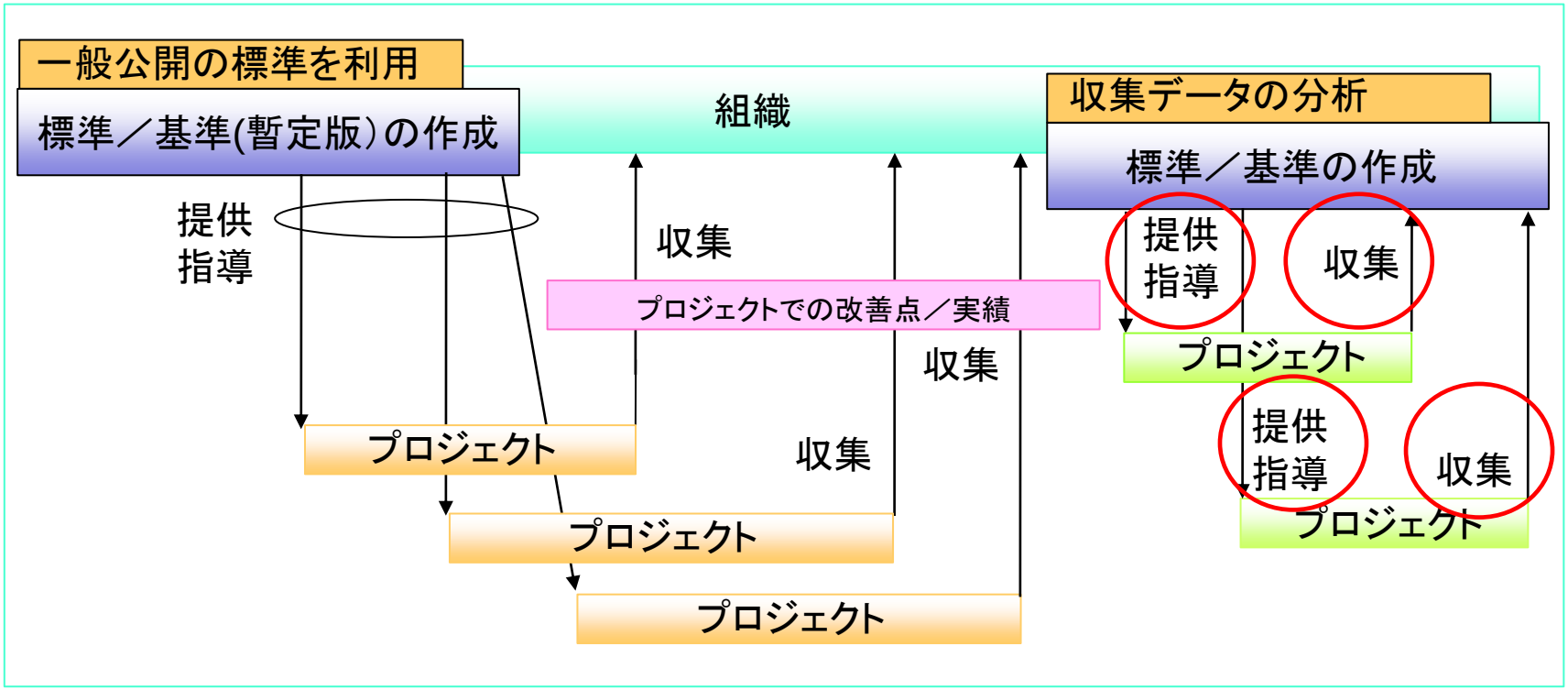


組織で収集する最低限のデータ、プロジェクトで適用する最低限の標準/基準を提供し、収集を行う。

データの収集には、ソフトウェアタグを利用し、組織で詳細化することが有効

5 組織レベルの改善

組織レベルの改善のサイクル



標準/基準の作成時には、より詳細化した指標を定義し、より精度が高く、ベンチマーキングが可能なデータの収集を制度化できるようにする。

6 改善の継続

制度化

組織レベルで標準/基準を制定しても、それが必ずしもプロジェクトで意図通りに運用されるとは限らない。

(a) プロジェクトの特性に合わない
- 高信頼性を要求され、組織で定めた標準/基準では対応できない。
- 開発規模が小さく、組織で定めた標準/基準を使うとコストが高くなる。

(b) プロジェクトマネージャの判断
- 単に使いたくない。
- 重要性を感じない。
- プロジェクトメンバに徹底させるのが面倒。

(c) 標準/基準が不備
- 組織で定めたものに不備が多く、使うときに書き直しが多い。
- テーラリングするとほとんど別ものになる。

組織で継続的に標準/基準を徹底し、改善する仕組みを組み込んでおく必要がある。

6 改善の継続

環境の変化

制度化により標準/基準が定着しても環境の変化により、収集すべき指標が変化してくる場合がある。

(a) 技術の変化（例）

- COBOLでは、ファイル単位にプログラムが作成されたため、規模の計測とプログラム本数の計測は単純に行う事ができた。
- Javaでは、クラス、メソッドがファイル単位とは限らず、重複を排除するなどの処理によって、規模の計測結果が異なる。
- 静的解析による警告への対応で、バグの削減が可能である。
- XML, XSLT, XSD, JSP, JavaScript, XHTML, CSS等、稼働させるための構成要素の種類が増えている。
- Webではデザイン要素も多く、単純に計測できない。

6 改善の継続

環境の変化

制度化により標準/基準が定着しても環境の変化により、収集すべき指標が変化してくる場合がある。

(b) インフラの変化（例）
- クラウドのためのインフラ
- オープンソース
- 性能の向上
- パッケージ製品（SAP、Oracle EBS等の適用）

(c) 法律
- 法律による各種制限事項の増加

見積り条件へのインパクトが大きく、常に見直しが必要

↓

組織レベルで継続的な改善