定量的ソフトウェア開発管理
Quantitative Management of Software Projects
（１－２）

ソフトウェア開発における定量化・計測
Measurement in software development

門田暁人 Akito Monden
monden@okayama-u.ac.jp

ソフトウェア開発における定量化・計測

- ソフトウェア開発（進捗状況，コストなど），ソフトウェアそのものの属性（例：信頼性，品質，使い勝手，保守性）を評価することは難しいが，計測によって少しでも理解を深めることが重要
- データ（計測結果）をもとに管理する方が何もないより良い．
- ただし，計測の目的は，管理者，開発者，ユーザが何を知る必要があるのかということに依存して，個別に設定しなければならない．

例

- 管理者
  - 各作業のコストはどれくらいか？
  - 開発者の生産性はどのくらいか？
  - 開発されたコードはどれくらい良いか？
  - ユーザは開発された製品に満足するか？
  - どうすれば残業を減らせるか？

答えの例

- 管理者
  - 各作業のコストはどれくらいか？
    - 作業時間数×作業単価
  - 開発者の生産性はどのくらいか？
    - コード行数÷時間，ファンクションポイント÷時間
    - バグ１件あたりの平均修正時間
    - 1時間あたりのテストケース実行数
  - 開発されたコードはどれくらい良いか？
    - 出荷後バグ密度，コードクローン含有率， C&Kメトリクス
  - ユーザは開発された製品に満足するか？
    - ユーザアンケート，非機能要件の評価，類似製品との比較
  - どうすれば残業を減らせるか？
    - ・・・

計測により残業を減らした事例

- 出典:
  - パナソニックアドバンストテクノロジー(株)川崎雅弘さま
  - 於:SPJ Japan2007(ソフトウェアプロセス改善カンファレンス)
  - 「メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例」
- 内容:
  - 日々バグ対応に追われ，慢性的工数不足のプロジェクトに対し，課題を明らかにするために実証データ(メトリクス)収集を実施した.
  - 分析結果に基づいたプロセス改善を行い，工数削減を実現し，残業時間も大きく減った.

川崎雅弘,"メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例",SPJ Japan2007

メトリクス収集方針

- 状況：日々バグ対応に追われ，慢性的工数不足
  - 何が問題か明確でないため，日々の作業に流されてしまう．
  - おそらく問題は．．．
    - 原因不明のまま長時間，延々と調べている．
    - 人による能力差が大きい．
    - 他部署で対応すべきバグが多い．
    - 他部署で対応すべきバグを深追いしている．
  - メトリクス計測により，問題を定量的に明らかにしたい．
- １週間だけメトリクスを収集する．
  - Excelシートに作業項目・時間を記録する．
  - 共有ディレクトリにExcelシートを置き，一定時間書き込みがない場合には自動的にメールで警告する．
  - 修正したバグ以外でも障害管理表に名前を記載する．

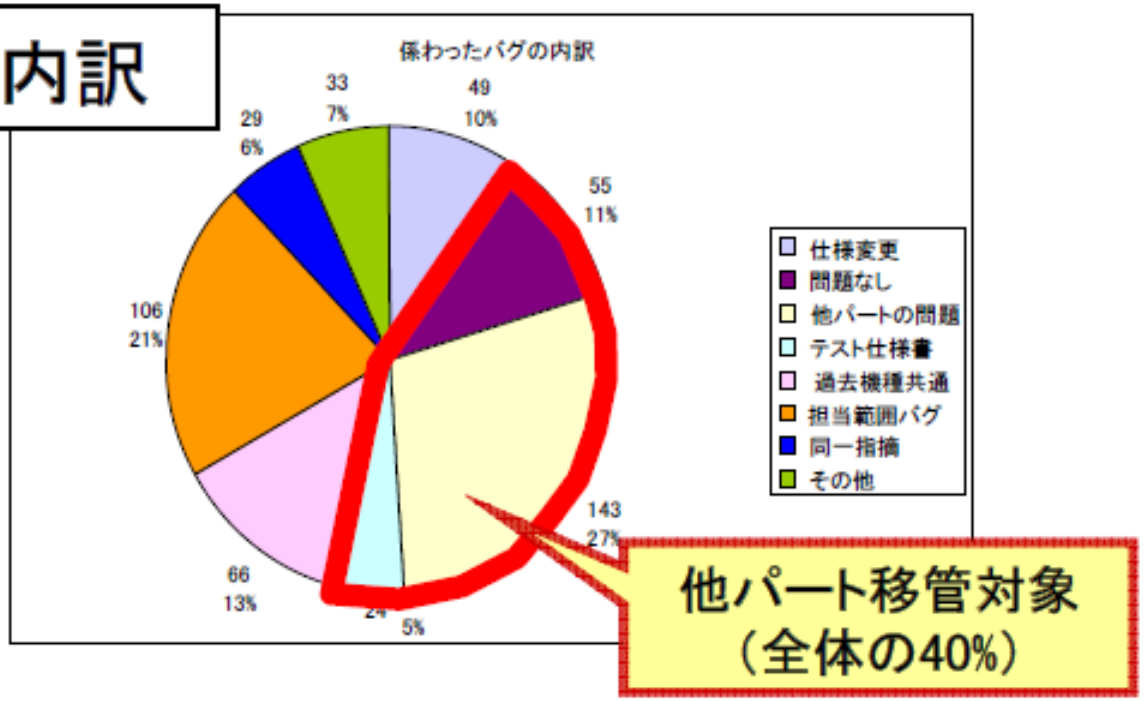
川崎雅弘,”メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例”,SPJ Japan2007

結果

- 分析結果
  - 平均バグ対応時間　22時間／件
  - 個人差は大きい．若手が長時間悩んでいる．
  - 約40%のバグが他部署に移管されている．

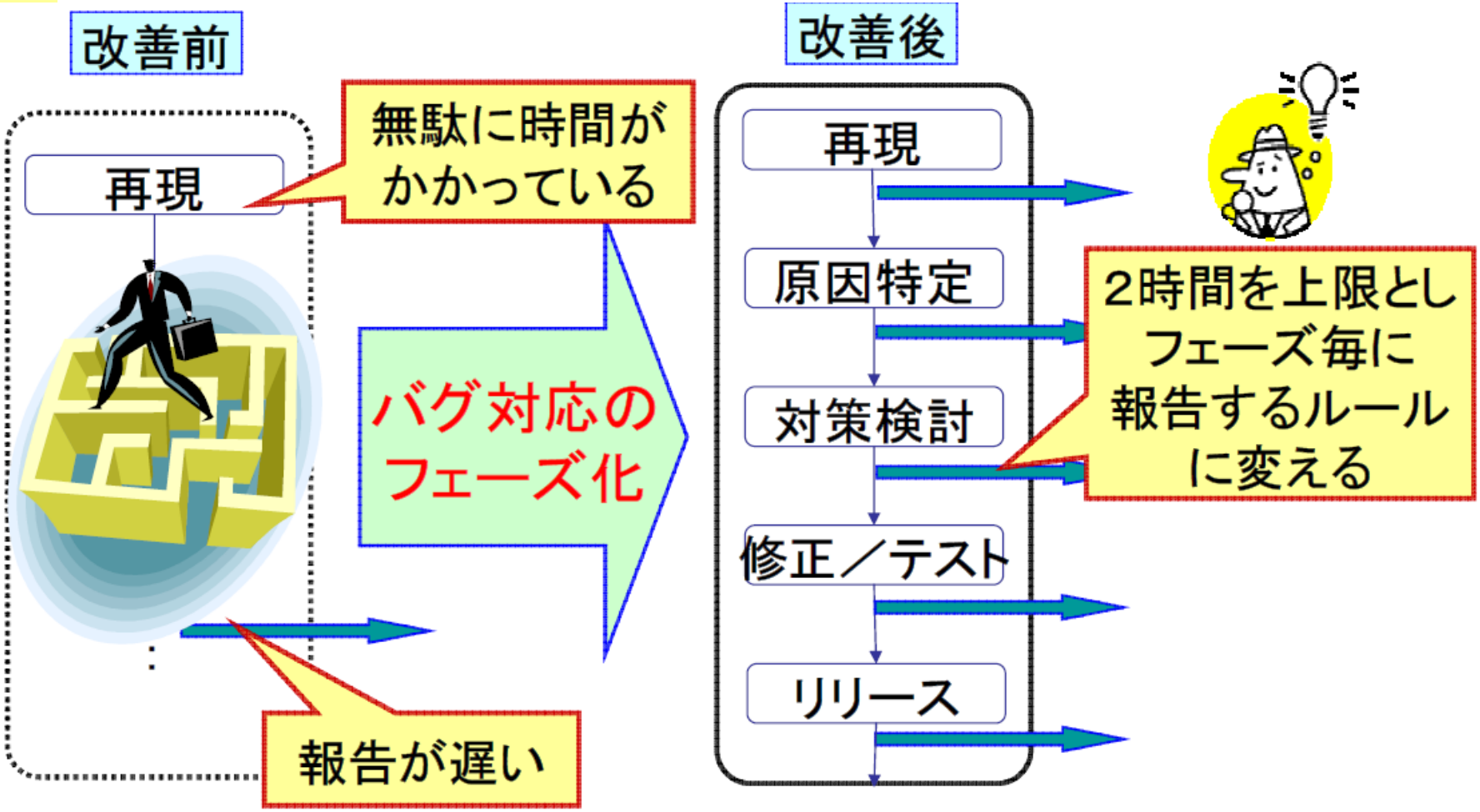
開発者が思っていたより
はるかに大きな数値

バグの内訳



川崎雅弘,”メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例”,SPJ Japan2007

改善策①



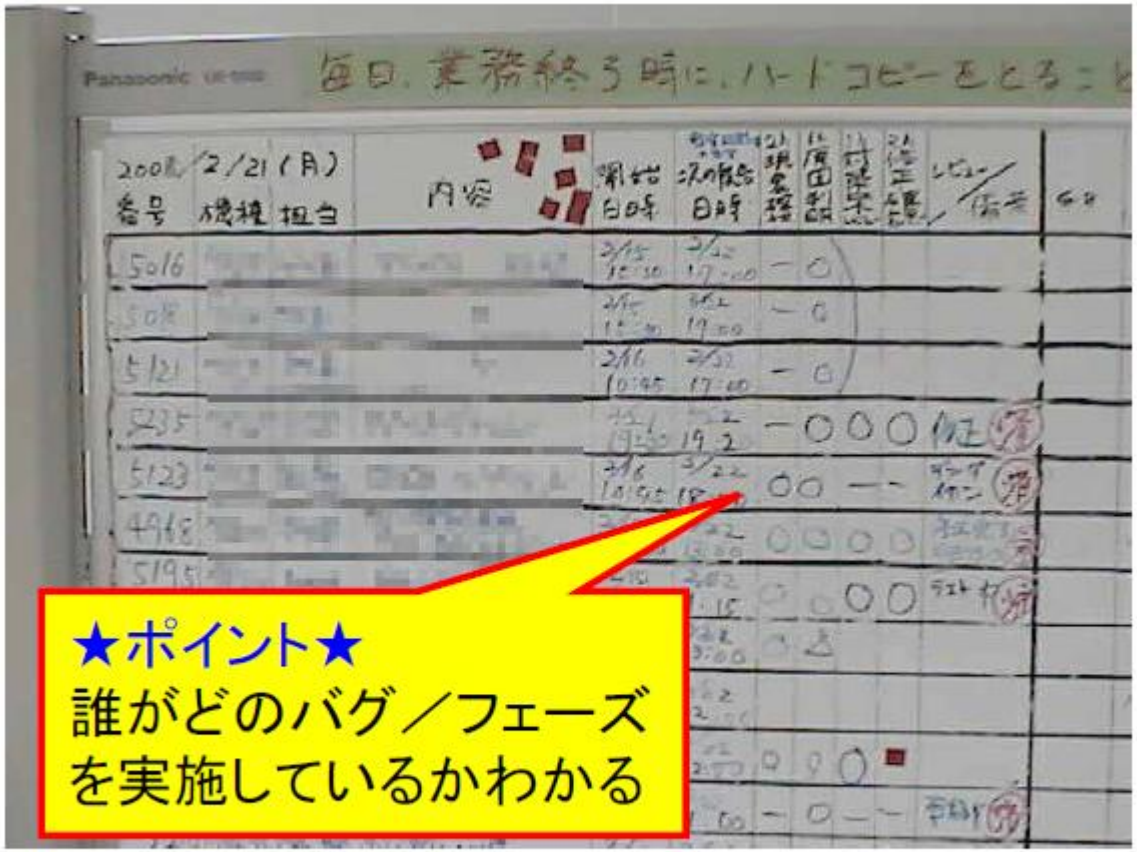
川崎雅弘,”メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例”,SPJ Japan2007

改善策②

・バグ対応フェーズ毎の時間上限を超えた場合、リーダに報告／相談
　→　適切なアドバイスをすることで対応が加速。特に若手に効果大

川崎雅弘,”メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例”,SPJ Japan2007

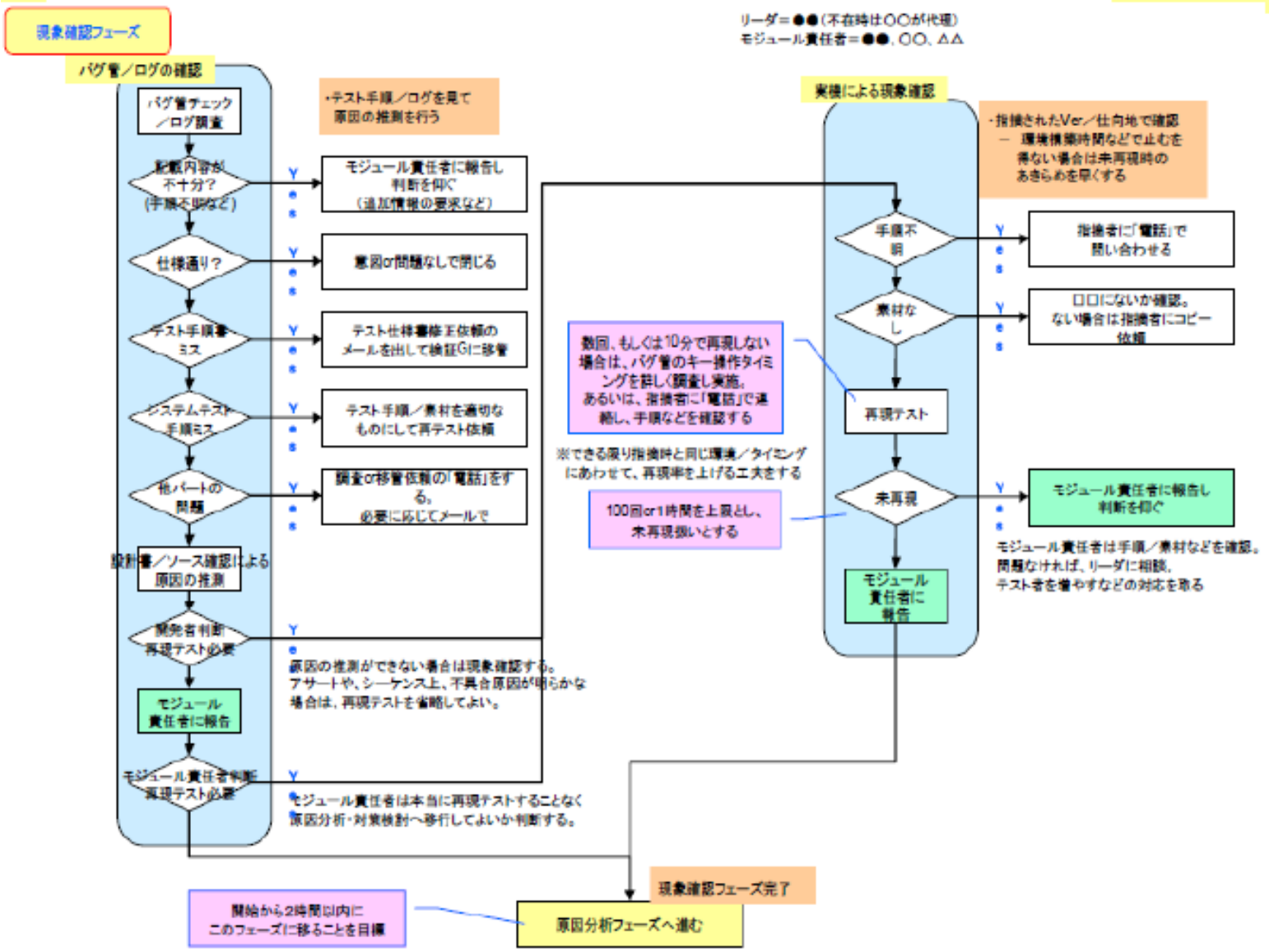
改善策③



バグ対応のフェーズ化と
ホワイトボードによる見える化

川崎雅弘,”メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例”,SPJ Japan2007

改善策④バグ対応手順の文書化



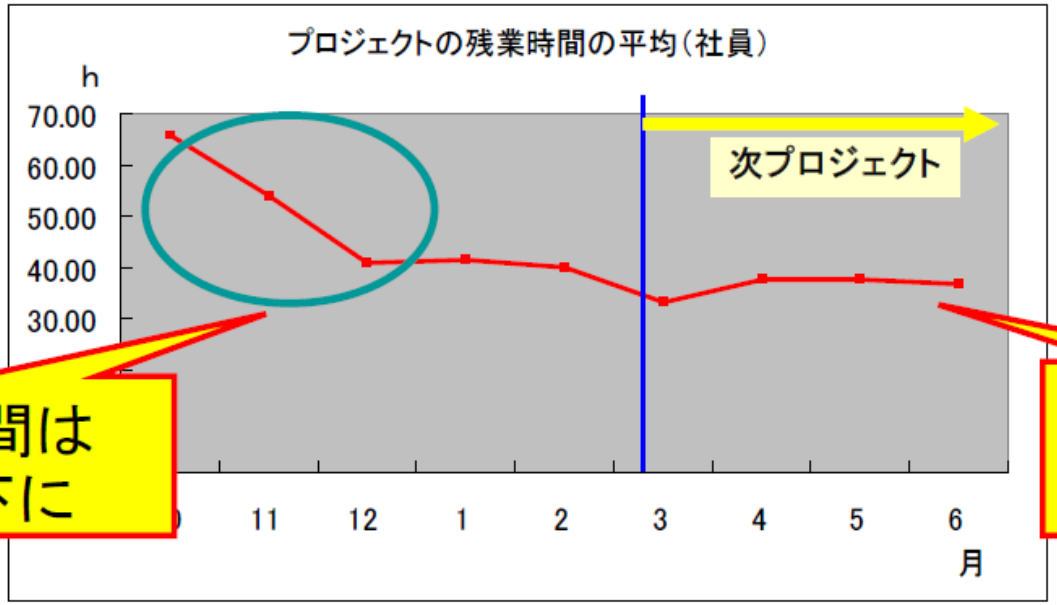
川崎雅弘,”メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例”,SPJ Japan2007

改善の効果

バグ平均対応時間
22時間／件 → 7.1時間／件

68%減の効率化

トータル工数が約10%削減。慢性的工数不足からの脱出



残業時間は
2/3以下に

次プロジェクト
では安定

川崎雅弘,”メトリクス活用による現場のプロセス改善～バグ対応工数の削減事例”,SPJ Japan2007

ソフトウェアメトリクス（Software metrics）

- ソフトウェアとその開発／利用過程を対象とした尺度
- 例
  - ソースコード行数（LOC; Lines Of Code）
  - ファンクションポイント（機能量）
  - サイクロマティック数（プログラム中の分岐の数+1）
  - コードクローン含有率（冗長なコードの含有率）
  - モジュール間結合度
  - 作業工数（1人月=20人日=160人時）
    - 開発コスト＝工数×人員単価
  - 発見バグ数，リリース後の故障数
  - バグ重要度分類基準

尺度水準

- 名義尺度（nominal scale）
  - 名前を対象に割り振る．もしくは数字を単なる名前として対象に割り振る．2つの対象に同じ数字が付いていればそれらは同じカテゴリに属する．
  - 例：開発種別＝1…新規，2…改修・保守，3…再開発，4…拡張
- 順序尺度（ordinal scale）
  - 対象に割り振られた数字は測定する性質の順序を表す．数字は等しいかどうかや，順序（大小）による比較が可能
  - 例：要求仕様の明確さ
    1. 非常に明確
    2. かなり明確
    3. やや曖昧
    4. 曖昧

尺度水準

- 間隔尺度（interval scale）
  - 対象に割り振られた数字は順序水準の性質を全て見たし，さらに差が等しいということは間隔が等しいということを意味する．
  - 例：開発年
- 比例尺度，比尺度（ratio scale）
  - 対象に割り振られた数字は間隔尺度の性質を全て見たし，さらにその中のペアの比にも，乗除の演算にも意味がある．
  - 例：ソースコード行数，開発期間，ファンクションポイント

ソフトウェアメトリクスの分類

- プロダクトメトリクス
  - ソースコード行数，バグ数など
- 資源メトリクス
  - 開発者数
  - 作業時間
  - 開発者の能力
- プロセスメトリクス
  - 進捗状況，作業効率，生産性

ソフトウェアの規模

- 最も基本的な属性の一つ
- 2種類の規模
  - 物理的（解決側，内部）：LOC
  - 機能的（問題側，外部）：ファンクションポイント，フィーチャーポイント

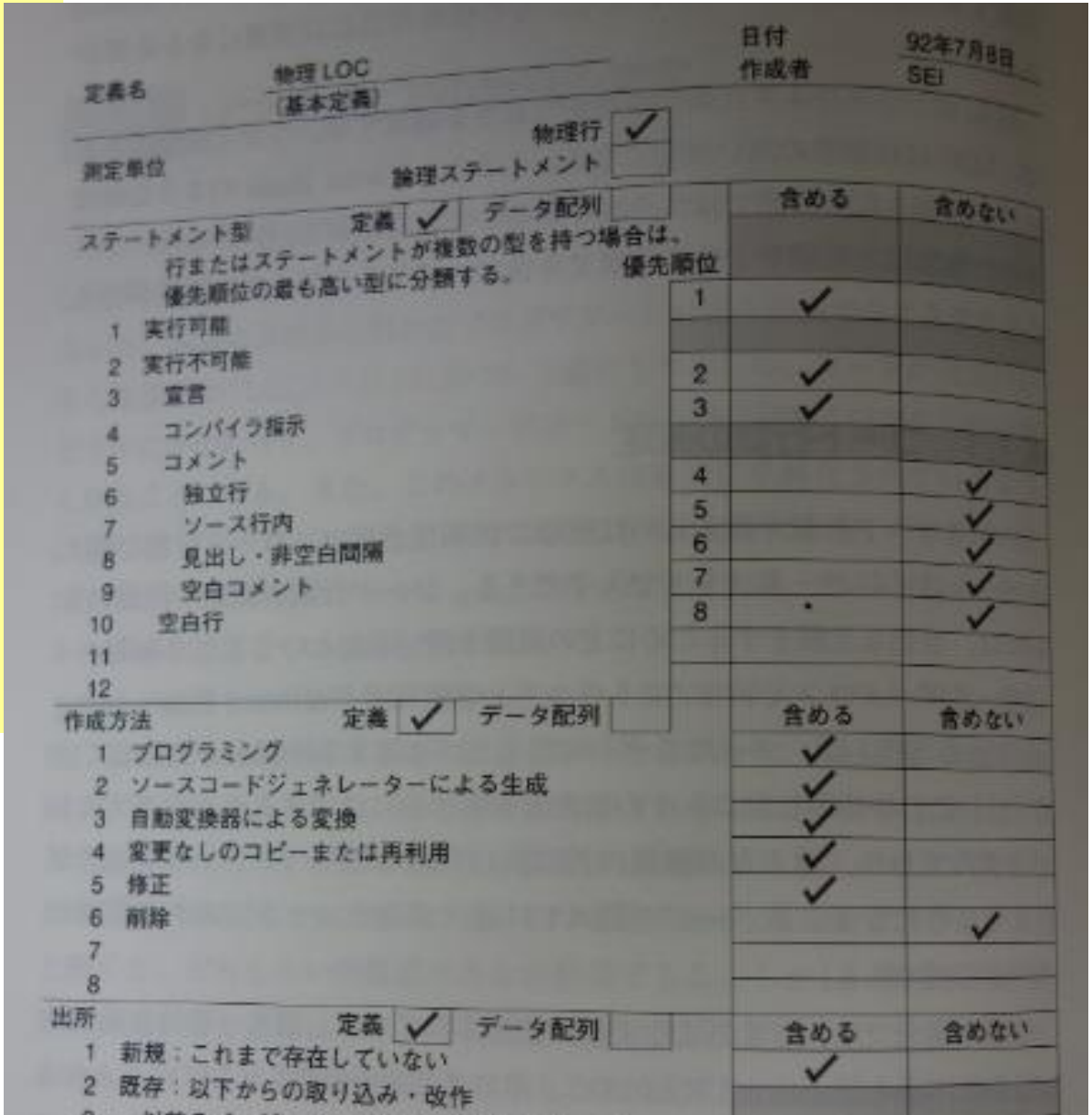
ソースコード行数(SLOC, LOC)

- 単に数えることは簡単
- 目的に合った測定規則を用いる.
  - コメントは？
  - ;のみの行は？
  - 自動生成されたコードは？
- 一般的な規則
  - NLOC(非コメント行数・有効行数):コメントのみの行と空行を除く行数
  - LLOC(論理行数):コメント行, 空行, カッコのみの行をカウントしない, N個のステートメントをN行とカウントする.

CMU/SEIコード数え方チェックリスト

- 何を計測対象に含めるかを明確にする.
  - コンパイラ指示文 #define，#ifdefなど
  - 自動生成コード
  - 流用コード
  - 商用ライブラリ
  - 自前のライブラリ
  - . . .

CMU/SEIコード数え方チェックリスト



定義名　物理 LOC（基本定義）　日付　92年7月8日
作成者　SEI
測定単位　物理行 ✓　論理ステートメント

ステートメント型　定義 ✓　データ配列
行またはステートメントが複数の型を持つ場合は、優先順位の最も高い型に分類する。

		優先順位	含める	含めない
1	実行可能	1	✓	
2	実行不可能			
3	宣言	2	✓	
4	コンパイラ指示	3	✓	
5	コメント			
6	独立行	4		✓
7	ソース行内	5		✓
8	見出し・非空白間隔	6		✓
9	空白コメント	7		✓
10	空白行	8	・	✓
11				
12				

作成方法　定義 ✓　データ配列

		含める	含めない
1	プログラミング	✓	
2	ソースコードジェネレーターによる生成	✓	
3	自動変換器による変換	✓	
4	変更なしのコピーまたは再利用	✓	
5	修正	✓	
6	削除		✓
7			
8			

出所　定義 ✓　データ配列

		含める	含めない
1	新規：これまで存在していない	✓	
2	既存：以下からの取り込み・改作		

出典：野中，鷲崎訳，
L.Laird, M.Brennan著
演習で学ぶソフトウェアメトリクスの基礎
p.50，日経BP，2009.

LOC計測時の留意点

- 再利用と修正
  - 生産性をみるためには再利用範囲の識別が必要
  - NASAの分類：修正なし再利用，25%未満修正，25%以上修正，新規
  - 経験則：25%を超える修正は，新規実装と同程度の工数
- 非手続き型言語のLOC
  - ビジュアル言語（Visual Basic等）：役立たない
  - オブジェクト指向言語（Java等）：利用されるが，手続き型言語ほどは役立たない．
- 言語の相違の考慮⇒言語補正係数

言語補正係数

- QSM社による調査（2005年）
- 1ファンクションポイントあたりの行数（中央値）
  - C　104
  - C++　53
  - COBOL　77
  - Java　59
  - アセンブリ　157