コードクローン検出技術とその応用

システム改善を目的としたクローン分析ツールの適用

Code Clone Analysis Applying to System Re- Structuring

大槻 繁(一)
永井昌子(CIJ)

CIJ Computer Institute of Japan

2007.2.7, 9
ICHI

お話の概要

システムの
見積り評価
コスト分析

ソースコードを対象とした
ソリューション

ソフトウェアの
開発•保守

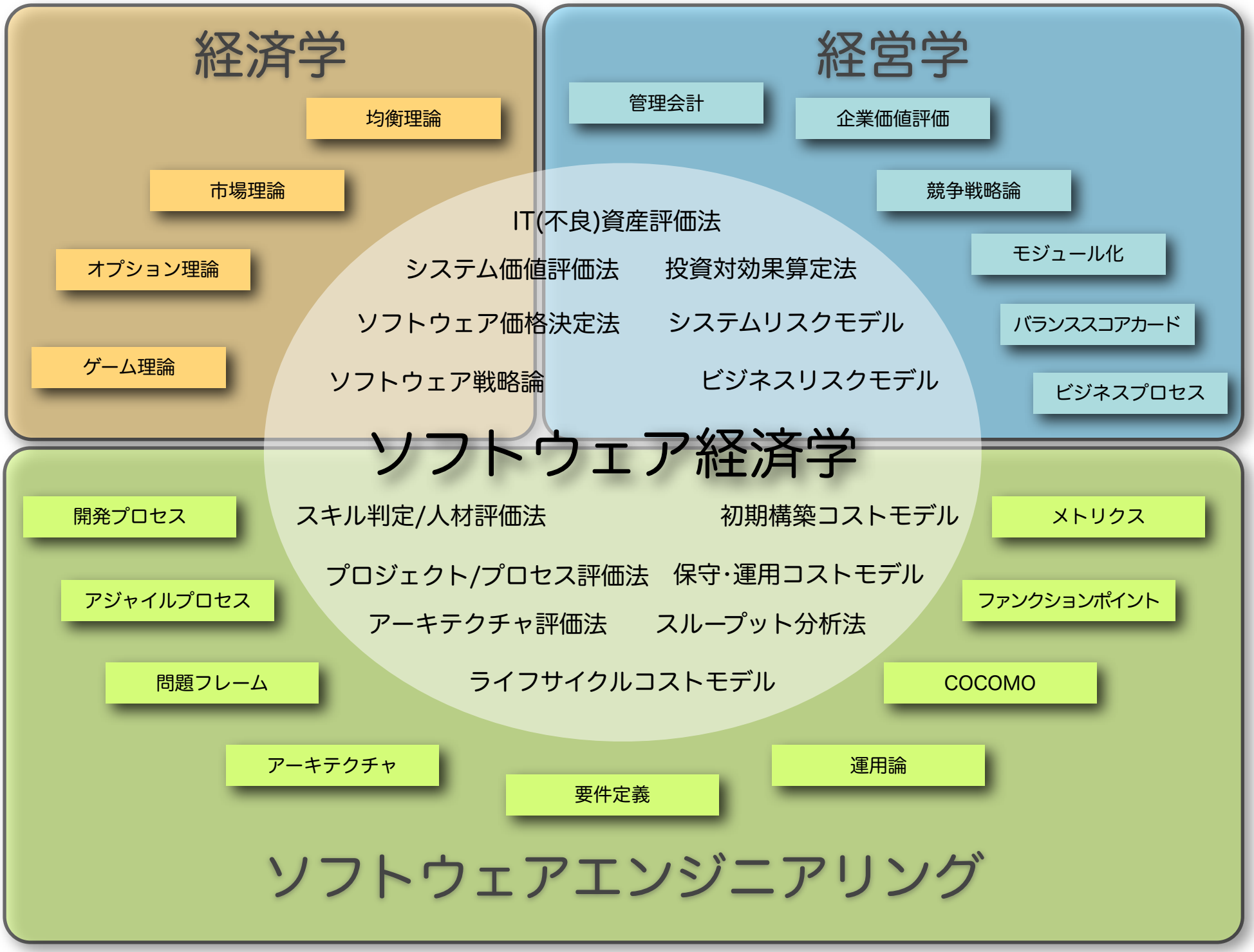
- コードクローン検出技術応用の可能性（大槻）
  - ソフトウェア経済学
  - IT資産評価
  - リファクタリング
- クローン分析ツール適用（試用）の実際（永井）
- 自由な討論

『ソフトウェア経済学』が明らかにしようとしていること

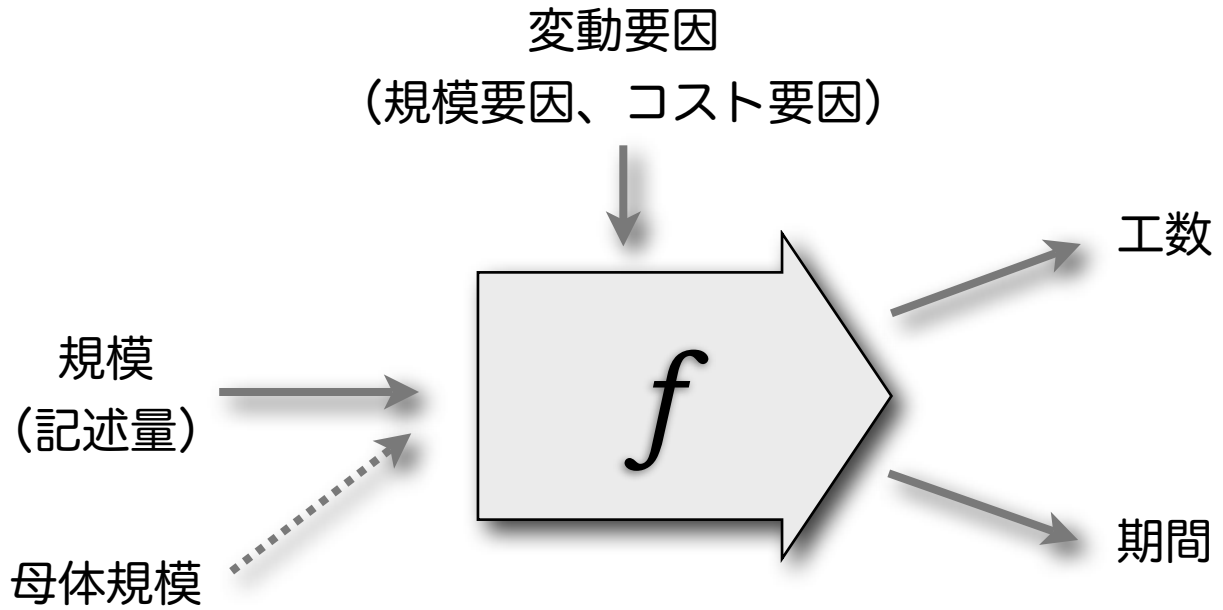
- ソフトウェア、システム、サービス等の無形財の利用、開発、保守、運用、破棄の総合的な社会/経済的な振舞い
- 市場、組織、部門、プロジェクト、チーム、個人の一貫した社会/経済的な振舞い
- 価値、価格、費用（コスト）の定式化と、これ間の関係

例：コードクローン検出技術、ソースコード解析、リファクタリング等の適用によってシステムの価値がどの程度上がるのか、コスト削減がどの程度図れるのか？

ソフトウェア経済学


経済学
均衡理論
市場理論
オプション理論
ゲーム理論
経営学
管理会計
企業価値評価
競争戦略論
モジュール化
バランススコアカード
ビジネスプロセス
IT(不良)資産評価法
システム価値評価法
投資対効果算定法
ソフトウェア価格決定法
システムリスクモデル
ソフトウェア戦略論
ビジネスリスクモデル
ソフトウェア経済学
開発プロセス
スキル判定/人材評価法
初期構築コストモデル
メトリクス
プロジェクト/プロセス評価法
保守･運用コストモデル
アジャイルプロセス
ファンクションポイント
アーキテクチャ評価法
スループット分析法
問題フレーム
ライフサイクルコストモデル
COCOMO
アーキテクチャ
運用論
要件定義
ソフトウェアエンジニアリング

COCOMO(COnstructive COst MOdel)



- ソフトウェア開発が人間による「記述」の活動であるという普遍的な考え方に基づいている
- 記述の規模から、工数、および、開発期間を算出する関数（算術式）を提示している
- 規模が大きくなるとマネジメントオーバヘッドを生じ、コストが増大することを考慮している
- 開発に関する変動要因を２２種（規模要因５種、コスト要因１７種）を定義しており、的確にコスト分析に反映できる

大きなもの程、工数と期間がかかる！
とりわけ、母体規模が大きくて、複雑なものは大変

母体がある場合のコストモデル

V 規模、E 工数、Q 品質（コスト要因）、P 期間、f コスト関数
とした場合、COCOMOは、次のように定式化できます。

$$E = f_E(V,Q) \qquad P = f_P(V,Q)$$

これを母体がある場合に拡張すると、

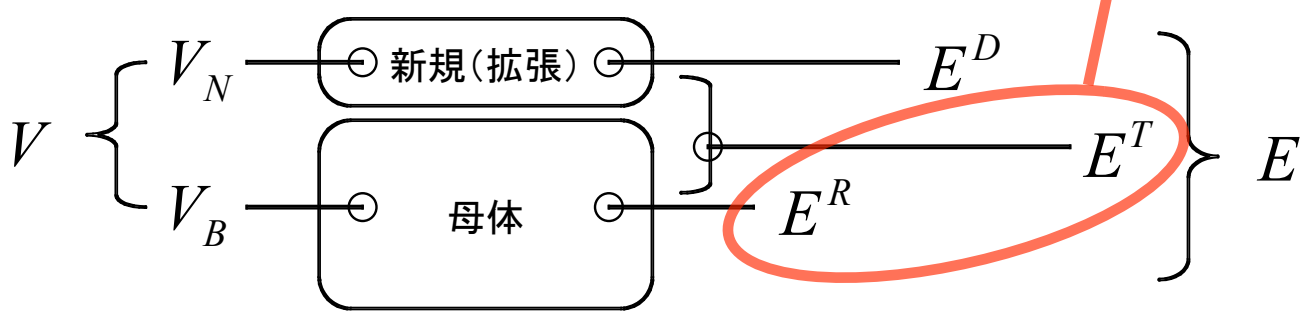
V_B 母体規模、 V_N 新規（拡張）規模、 E^R 理解工数、 E^D 開発工数
E^T 全体調整（統合テスト）工数、

$$V = V_B + V_N$$

$$E = E^R + E^D + E^T$$

$$= f_E^R(V_B,Q) + f_E^D(V_N,Q) + f_E^T(V,Q)$$

保守•改変時のボトルネック！



《規模の世界》 《工数の世界》

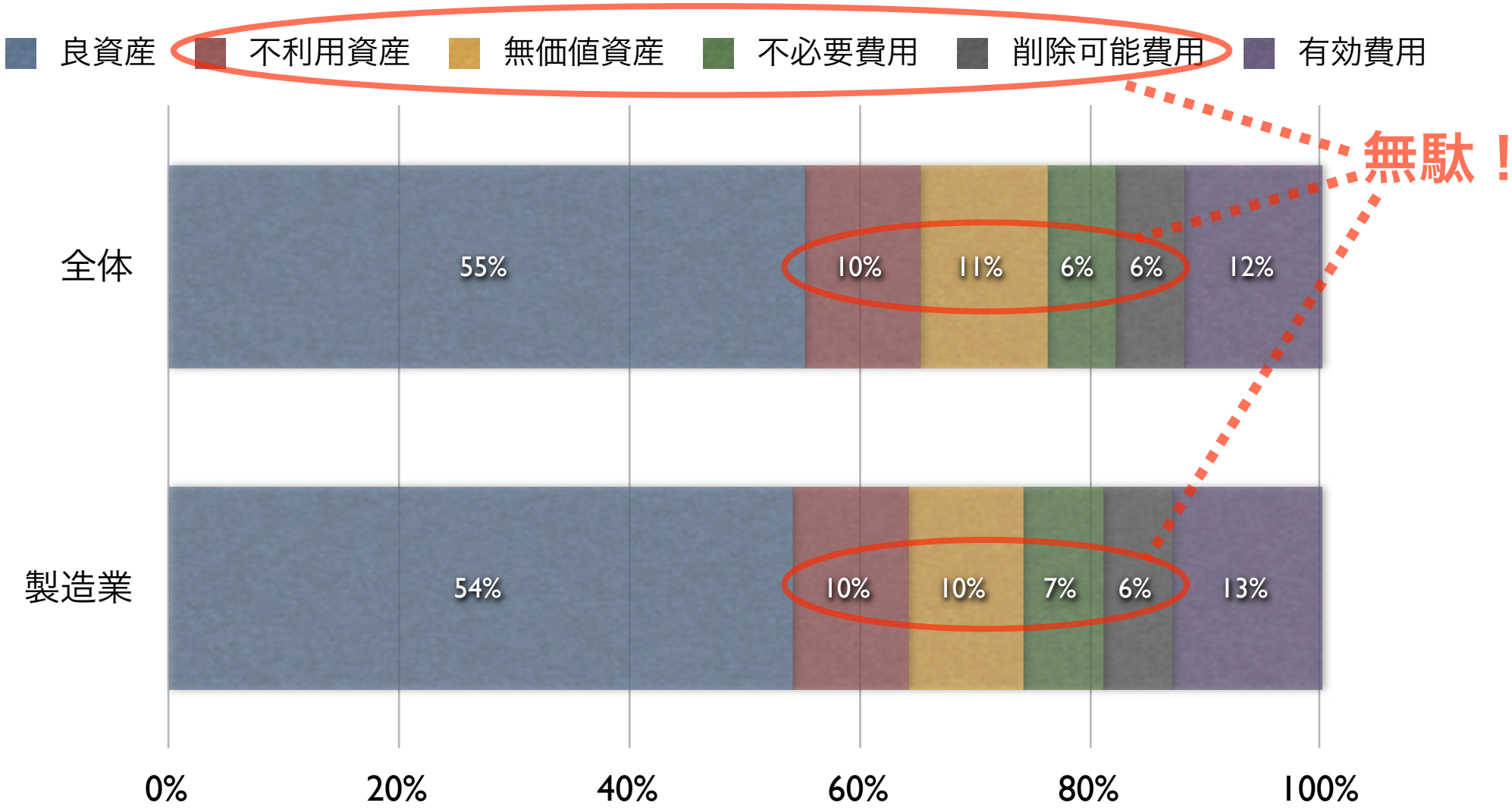
ソフトウェア開発における施策

- 良構造化：不良資産を破棄し、ソフトウェア資産（母体）を良構造に変換・保持すること
- 自動化：ツール活用やプロセスの最適化によって、誤りを減らし、作業を効率化すること
- 抽象化：人間の知的活動に使う言語や開発環境の抽象度を上げること

IT不良資産とは

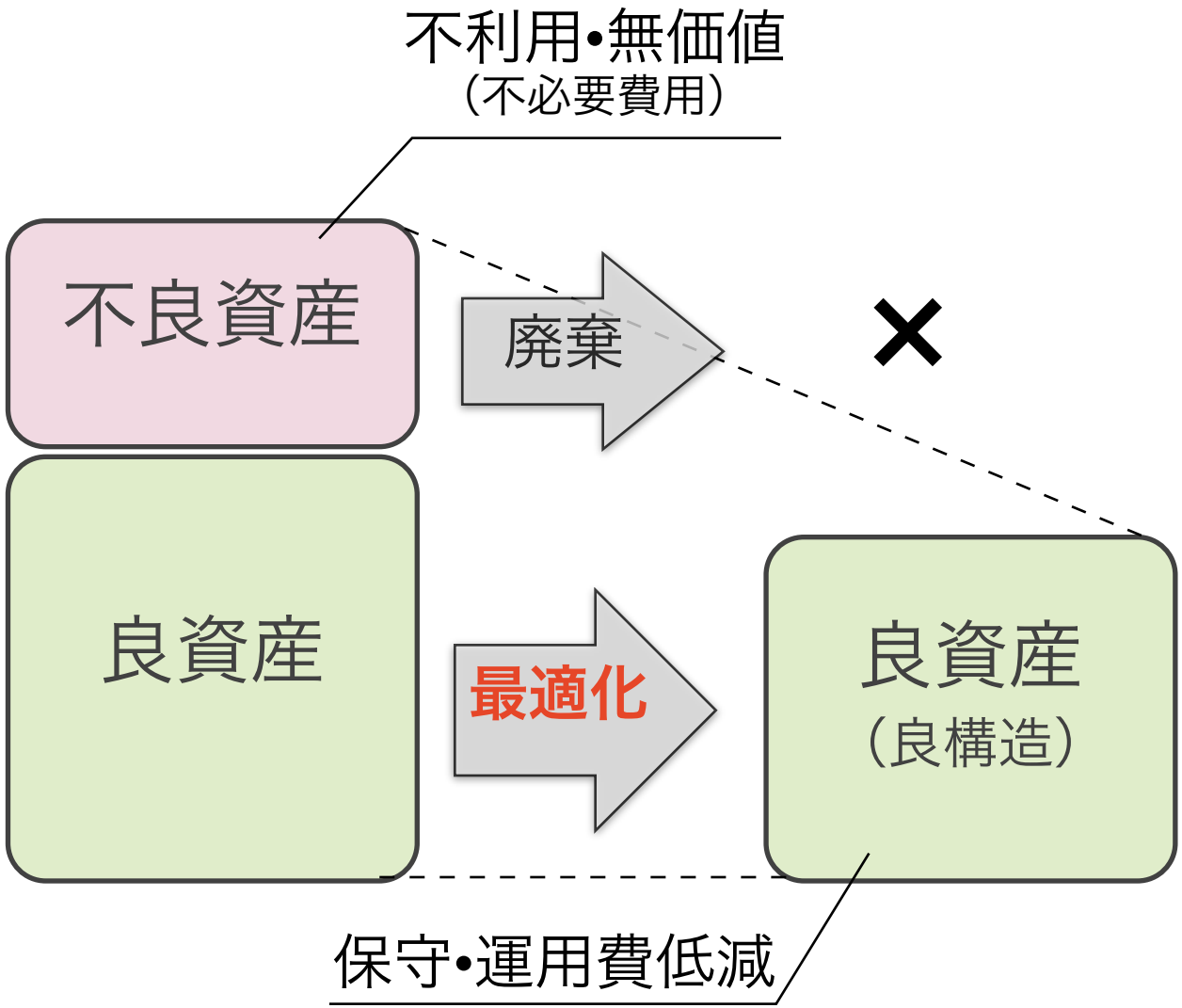
- 不良資産の種類
  - 不利用資産：ユーザが利用していないもの
  - 無価値資産：投資金額に見合った効果がないもの
  - 不必要費用：不利用資産、無価値資産を維持するための費用
  - 削除可能費用：低減できる費用、無駄、余剰人件費
- 不良資産化の原因
  - 経営層の戦略•ビジョンの欠如
  - 経営層とシステム部門ととの乖離
  - 予算消化型の投資
  - システム部門とユーザ部門との乖離
  - ベンダー、システム子会社との癒着

IT不良資産の実態

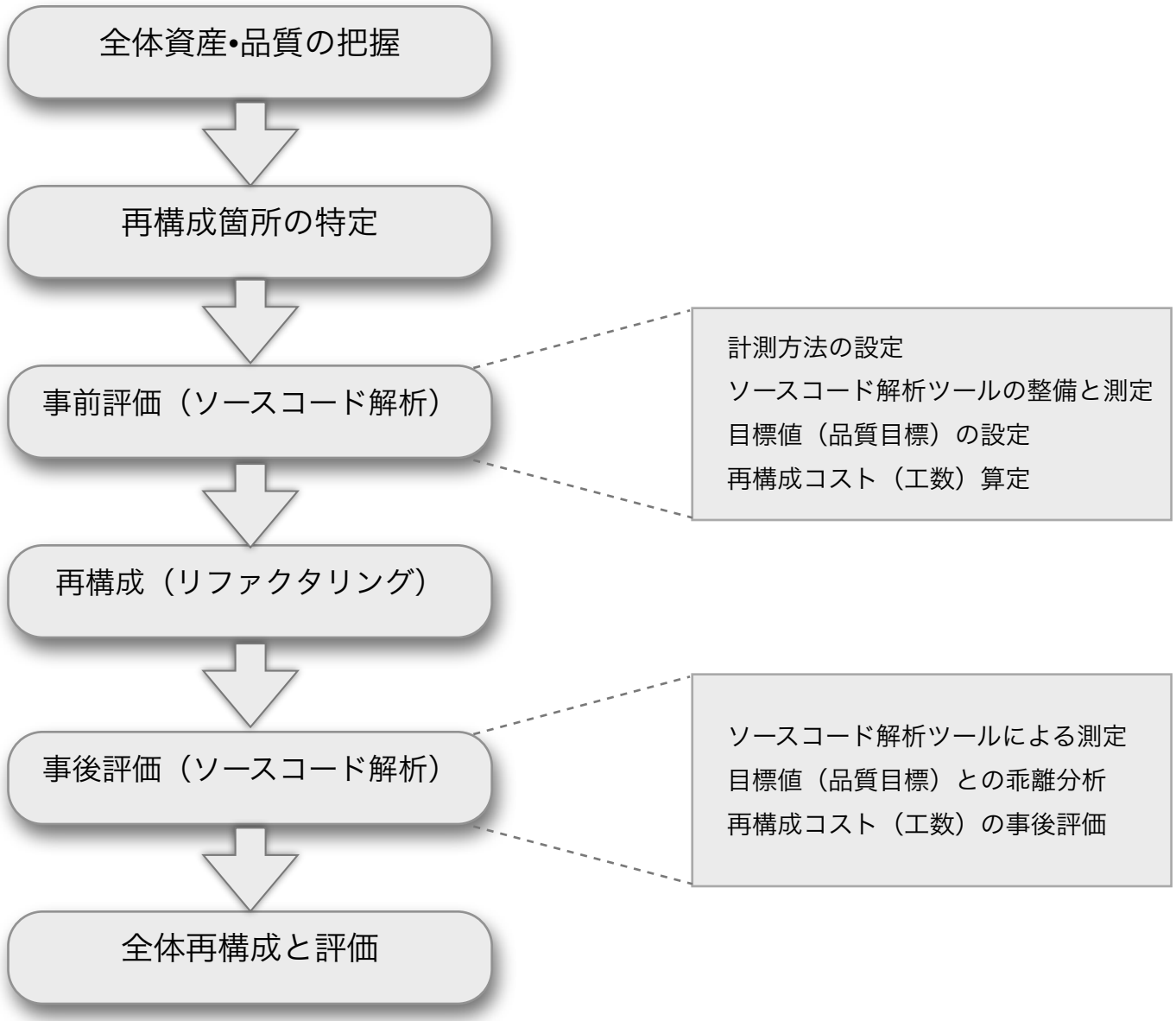


森 秀明, 「IT不良資産」, ダイヤモンド社, 2003

不良資産への対処



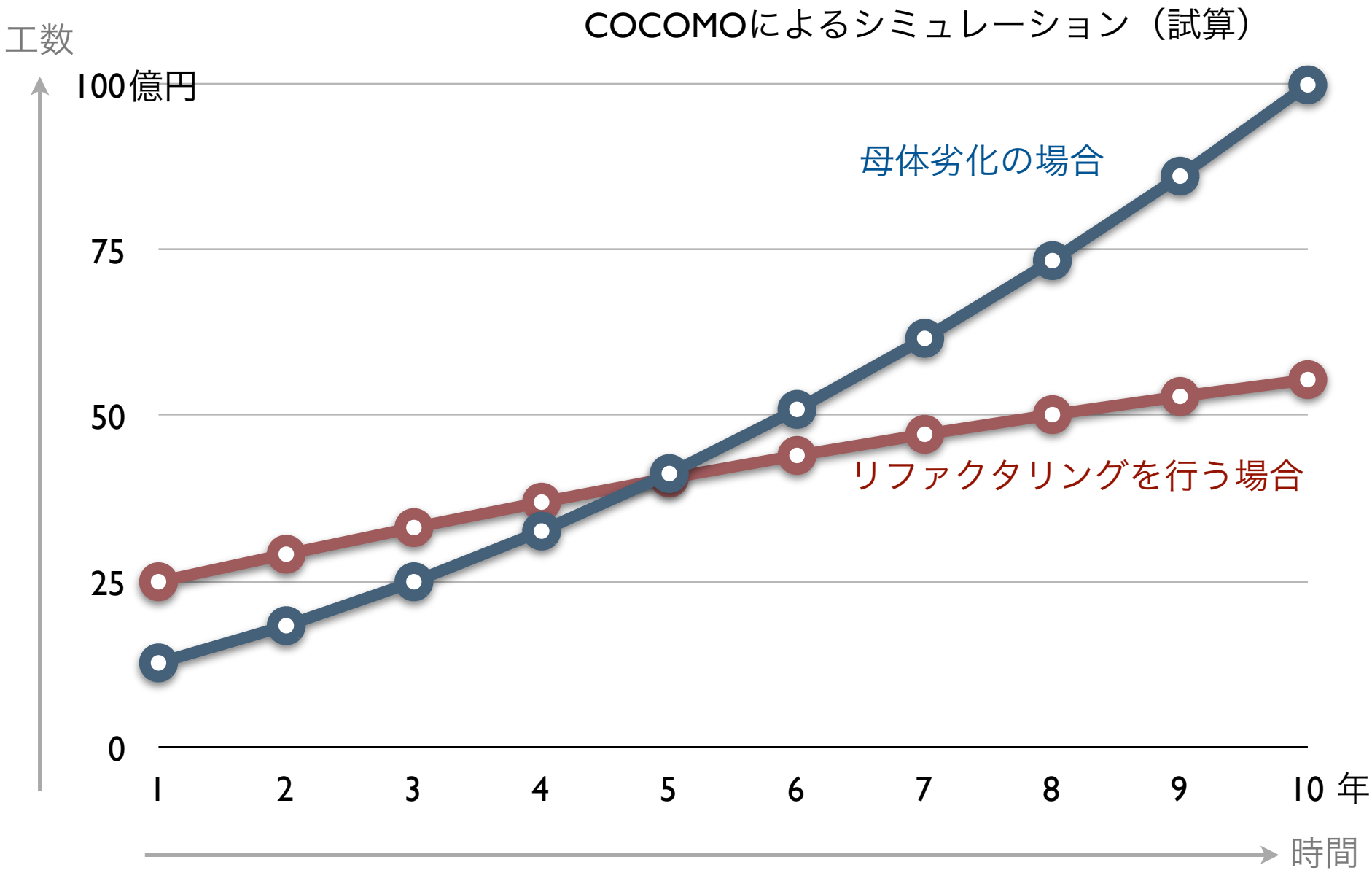
リファクタリング手順


全体資産•品質の把握
再構成箇所の特定
事前評価（ソースコード解析）
再構成（リファクタリング）
事後評価（ソースコード解析）
全体再構成と評価
計測方法の設定
ソースコード解析ツールの整備と測定
目標値（品質目標）の設定
再構成コスト（工数）算定
ソースコード解析ツールによる測定
目標値（品質目標）との乖離分析
再構成コスト（工数）の事後評価

ソースコード解析による品質測定

内部品質特性		ソースコード指標	
特性名	特性の説明	典型的指標	説明
一貫性	設計，製造の技法や表記法，用語などが統一されていること	表現順守率	コーディング規則などで規定される命名規則や文法に従っている文の比率
自己記述性	機能および機能間の関連が完結していること	コメント率	ヘッダー部の仕様記述に関する記述のコメント文の比率
データ共通性	データを内外のシステムと共通に使用できること	データ参照・被参照数	共通データの数，およびデータへの参照数
通信手順共通性	通信手順やインタフェースが共通化していること	モジュール参照・被参照数	通信やインタフェース・モジュールに対する参照・被参照数
アクセス可能性	プログラムの機能や関連装置を選択して自由に使用できること	特定モジュール参照数	プログラム機能や装置に関するモジュールの参照数
アクセス制御性	ソフトウエアやデータへのアクセスを制御できること	特定モジュール参照数	アクセス制御モジュールに対する参照数
アクセス監査性	ソフトウエアやデータへのアクセス記録を残せること	特定モジュール参照数	アクセス記録モジュールに対する参照数
堅固性	誤って操作しても，データやプログラムが破壊されないこと	前提条件充足率	入出力モジュールに対する前提条件の充足率
整合性	異常が発生してもデータやプログラムが破壊されないこと	実行条件充足率	モジュール全体に対する前提条件の充足率
モジュール性	ソフトウエアが構造化され，変更・修正などが局所的に済むこと	結合度	モジュール間の依存の度合い
単純性	仕様の実現方法が簡単であること	規模，複雑度	規模はコード行数（LOC)とHalsteadのメトリクス，複雑度はMcCabeのメトリクス
計測性	プログラムの動作状況を観察，観測できること	特定モジュール参照数	観測するモジュールに関するモジュール参照数
自己包含性	他のプログラムに依存しないで機能を満たせること	凝集度	モジュールが機能，情報を一元的にカプセル化している度合い
統一性	意味，表現，手順が一義的，同一的であること	表現順守率	コーディング規則などで規定される命名規則や文法に従っている文の比率
簡潔性	表現が短く，明解なこと	不要コード率,クローン率	論理的に実行されないか，利用条件によって実行されないコード行数率、コピー/重複のある比率
動的効率性	動作の応答時間，処理時間，スループットがよいこと	ダイナミックステップ数	実行時に実行されるステップ数
資源使用性	実行する際に，使用する資源量や時間が少なくて済むこと	メモリー占有量	実行時に実行されるローディングされるモジュール，データの占有量
拡張性	仕様の追加，変更に対して，容易に対応できること	結合度，凝集度	モジュール性と自己包含性を参照
ソフトウエアシステム独立性	特定のＯＳ，コンパイラなどに依存しないこと	ＯＳ非依存モジュール率	OSモジュールを参照しないモジュール比率
マシン独立性	特定の機種，装置，端末などに依存しないこと	機種非依存モジュール率	機種依存モジュールを参照しないモジュール比率
データ独立性	特定のデータ，データベース管理システムなどに依存しないこと	DB非依存モジュール率	DBモジュールを参照しないモジュール比率
伝達性	プログラムの入出力形式や内容が使いやすく統一されていること	メッセージ表現順守率	メッセージに関する規則に順守しているメッセージ定義の比率

リファクタリングの効果



お話のまとめ

- ソースコードは主要な「記述」成果物。
  ソフトウェア開発の中心的活動は「記述」であり、
  「記述」の経済性/合理性を分析する必要がある
- 母体を凝縮すること（不良資産の削減、リファクタリング）は、開発•保守効率向上の要
- ソースコード解析（クローン分析を含む）は、資産情況を把握する有効な指標と分析方法を提供