ESS2013 チュートリアル資料 2013年10月15日

メトリクスによるソフトウェア品質の評価と改善

鷲崎　弘宜

早稲田大学グローバルソフトウェアエンジニアリング研究所

washizaki@waseda.jp

Twitter: @Hiro_Washi

鷲崎 弘宜（わしざきひろのり）
http://www.washi.cs.waseda.ac.jp/

再利用と品質保証を中心としたソフトウェア工学の研究と教育
- 早稲田大学グローバルソフトウェアエンジニアリング研究所 所長
- 早稲田大学基幹理工学部情報理工学科准教授
- 国立情報学研究所客員准教授
- 論文: ICSE, ICSR, ASE, ARES, Agile, SPLC, SEKE, PLoPなど

対外活動
- 日本科学技術連盟ソフトウェア品質管理研究会副委員長
- 電子情報通信学会ソフトウェアサイエンス研究会幹事
- ISO/IEC JTC1 SC7 WG20 国内委員会主査
- SEMAT Japan Chapter Chair
- IEEE CS Japan Chapter Secretary

執筆など
- 『演習で学ぶソフトウェアメトリクスの基礎』
- 『ソフトウェア品質知識体系 SQuBOK』
- 『ソフトウェアパターン入門 – 基礎から応用へ -』
- 『AspectJによるアスペクト指向プログラミング入門』ほか

本チュートリアルの目標

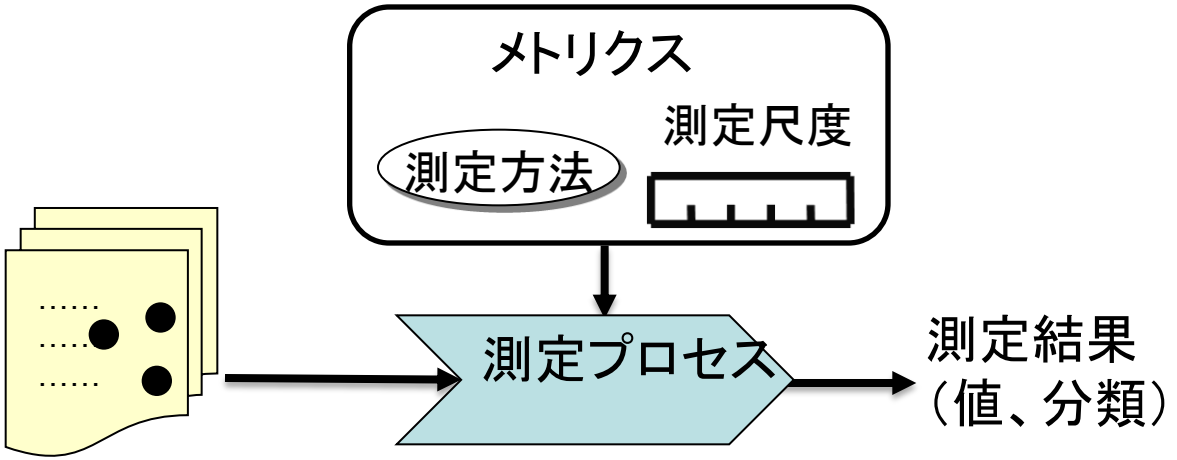
- メトリクスの基本概念と主要なもの、および落とし穴を理解する
- GQMの基礎を理解し、自分なりの価値観を得る
- 事例を通じてGQM適用に基づくソフトウェア品質の測定評価と改善の流れを理解する

目次

- **メトリクスの基本概念**
- 主要なメトリクス: 規模，複雑さ，欠陥
- GQMによるゴール指向な測定
- GQMを用いた組込みソフトウェア品質改善事例
- まとめ

メトリクス（Metric / Metrics）

- 測定の方法と尺度
  - 方法: 属性（測定可能な特徴）の尺度上の値や分類への対応付け
  - 尺度: 値や分類の集合
- 測定できない事柄は、管理できない（T. DeMarco）
  - ソフトウェア工学 = ソフトウェアの開発、運用、および保守に対する系統的で規律に基づいた定量的アプローチ [SWEBOK]
  - Prj成功率 31% → 定量的評価導入Prj 46% [矢口08]



[DeMarco]T. DeMarco, Software Engineering: An Idea Whose Time Has Come and Gone?, IEEE Software, 2009.
[SWEBOK] 松本吉弘 監訳: ソフトウェアエンジニアリング基礎知識体系—SWEBOK2004, オーム社, 2005.
[矢口08] 矢口竜太郎, 吉田洋平: 成功率は31.1%, 日経コンピュータ12月1日号, 2008.

何を測定するのか：測定対象のエンティティ

- プロセス
  - 入力を出力に変換するアクティビティ（活動）の集合
- プロダクト
  - アクティビティから出力される成果物や文書
- リソース
  - アクティビティの実施に必要な人、もの、金などの資源

N.E. Fenton, and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, PWS Publishing
野中誠 , ソフトウェア技術者のためのメトリクス基礎, 2010, TopSEセミナー

主なメトリクス

	名称	定義	用途
プロダクト	コード行数（LOC）	コードの行数	規模の把握、他のメトリクスの正規化
	ファンクションポイント（FP）	機能量	規模の見積もり
	サイクロマティック複雑度（CC）	制御フローグラフの経路数	複雑さの把握
	凝集度	モジュール内の要素のまとまり	複雑さの把握
	結合度	モジュール間の結合関係の多さ	複雑さの把握
プロセス	欠陥密度（DD）	欠陥数 / 新規・変更LOC	品質の把握、開発プロセス全体の有効性把握
	欠陥除去率（DRE）	欠陥摘出数 / 当該時点欠陥数	開発プロセス全体や工程の有効性把握
	消化済みテストケース数	当該時点までの消化済みテストケース数	テストの進捗把握

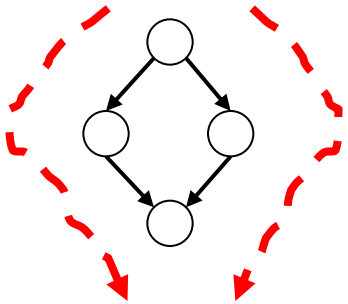
測定方法

- ソフトウェアの抽象的な特定側面を捉える
- 測定方法が異なれば測定結果も変わる

```
 1:/* strncat()は、文字列srcからcount数の文字を文字列destに付加し、さ
 2:らに終端にnull文字を付加する。重なるオブジェクト間でコピーしようとする
 3:場合、動作は未定義である。*/
 4:char *strncat(char *dest, const char *src, size_t count)
 5:{
 6:     char *temp=dest;
 7:     if (count) {
 8:          while (*dest)
 9:               dest++;
10:          while ((*dest++=*src++)) {
11:               if (--count ==0) {
12:                    *dest＝‘¥0’; break;
13:               }
14:          }
15:     } return temp;
16:}
```

例：サイクロマティック複雑度（Cyclomatic Complexity: CC）

- テキスト
  - プログラムの制御フロー上で他とは異なるリンク（エッジ）を持つ経路数
- ダイアグラム



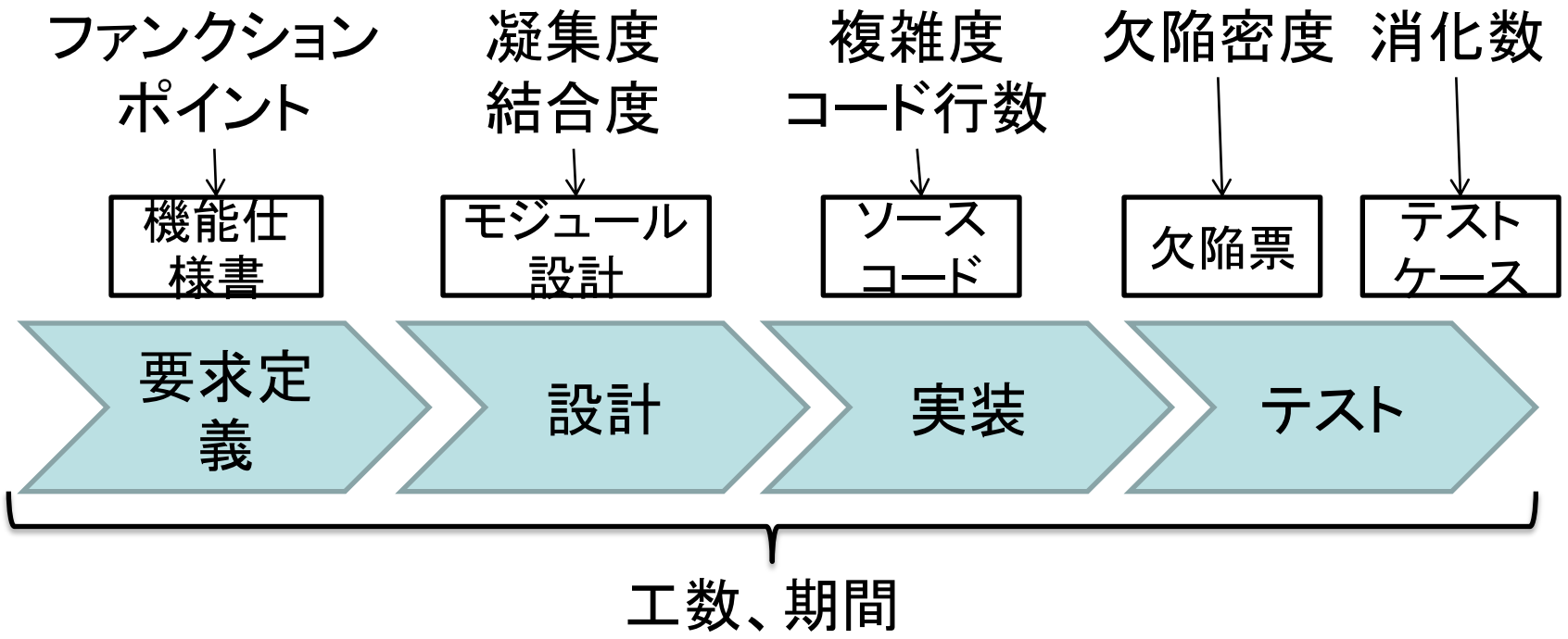
＝2

- アルゴリズム
  - V(g) = e – n + 2（ただし e: エッジ数、n: ノード数）
  - V(g) = bd + 1（ただし bd: 二分決定数）

出典: 野中誠 , 鷲崎 弘宜,ソフトウェア技術者のためのメトリクス基礎, 2010, TopSEセミナー
[Linda09] リンダ・M・ライルド, M・キャロル・ブレナン著, 野中誠, 鷲崎弘宜 訳 , "演習で学ぶソフトウエアメトリクスの基礎 ソフトウエアの測定と見積もりの正しい作法", 日経BP社 , 2009.

メトリクスの使いどころ

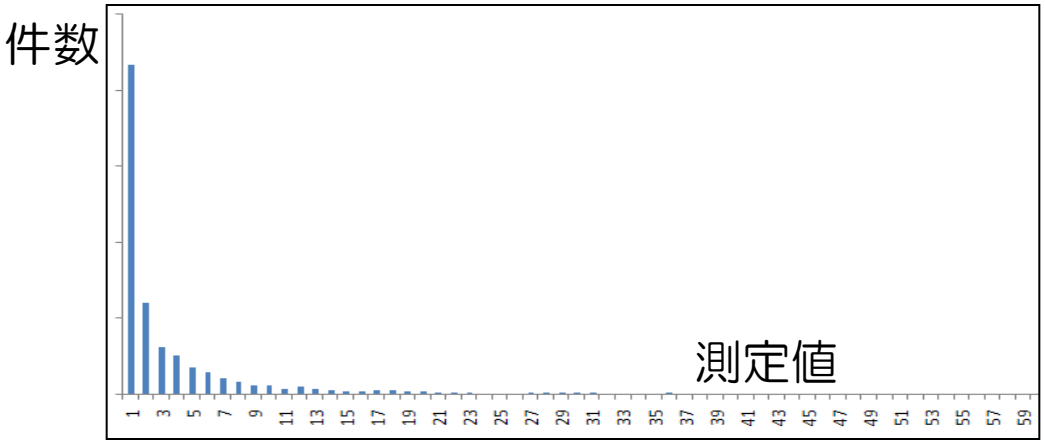
- 実態把握、問題の絞込み
  - 複雑さ、欠陥密度など
- 予測、計画
  - 工数見積もり、欠陥数の予測など

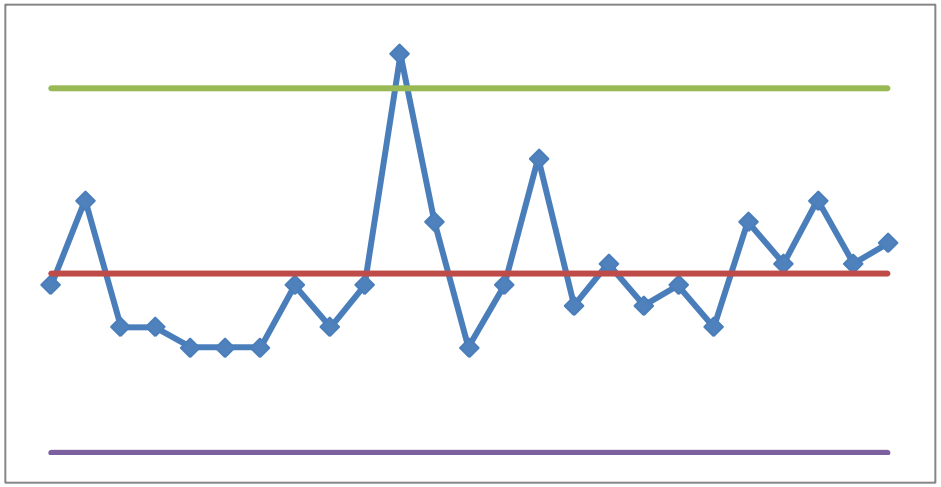


実態把握や問題絞込み

- ヒストグラム
  - 分布状況の把握
- 管理図
  - 平均や範囲のばらつき
  - 異常の特定

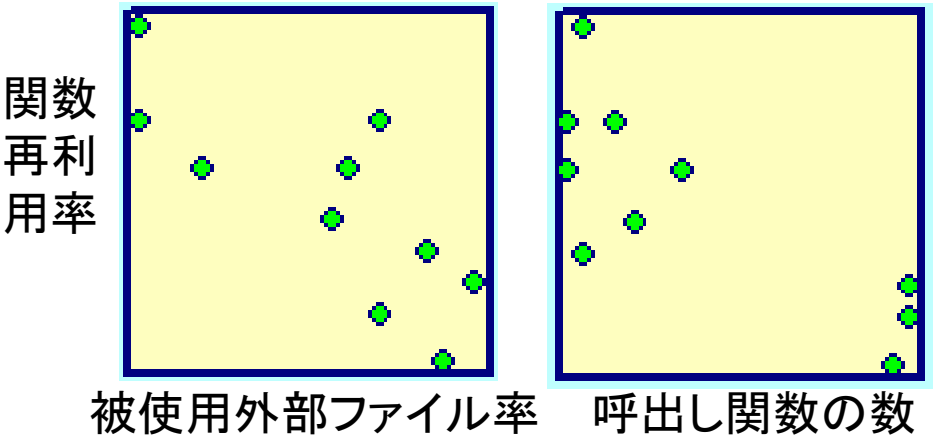
例: サイクロマティック複雑度



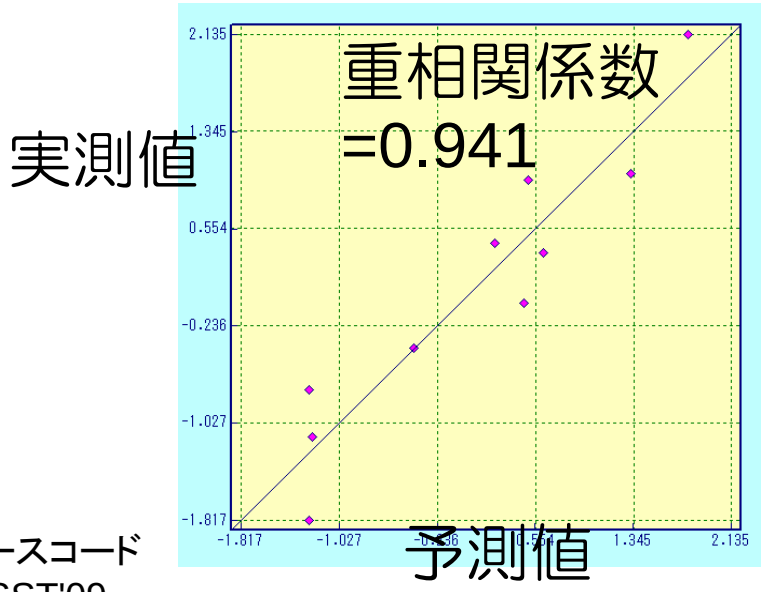


予測や計画

- 散布図
  - 関係の把握



- 関係性の分析
  - 単回帰分析: 1つの説明変数、1つの目的変数
  - 重回帰分析: 2つ以上の説明変数
  - 2値データのロジスティック回帰分析: 目的変数が0,1



鷲崎弘宜, 小池利和, 波木理恵子, 田邉浩之, "C言語プログラムソースコードの再利用性測定法とその評価", ソフトウェアテストシンポジウム JaSST'09

メトリクスの落とし穴

- 信頼性:「１件」「1行」が異なっている（かも）
  - 欠陥数、行数・・・
  - 客観性の高い測定、自動測定
- 妥当性:「実世界で測定」できることは「測定したい概念」の一部に過ぎない
  - 多面的な捉え方、現物確認
  - 測定・管理がプロジェクトに最重要とは限らない [DeMarco09]
- 負のホーソン効果
  - 本当に改善したいこと、目的指向

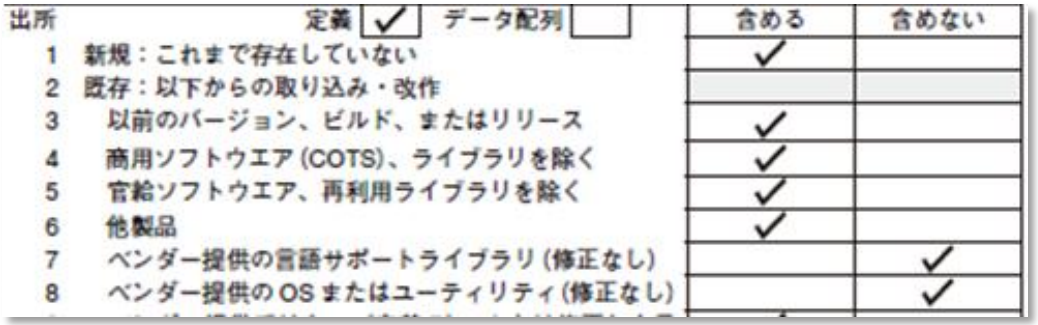
[DeMarco09]T. DeMarco, Software Engineering: An Idea Whose Time Has Come and Gone?, IEEE Software, 2009.

目次

- メトリクスの基本概念
- **主要なメトリクス: 規模，複雑さ，欠陥**
- GQMによるゴール指向な測定
- GQMを用いた組込みソフトウェア品質改善事例
- まとめ

規模: コード行数(LOC)の留意点

- 測定モデルの一貫性
  - NLOC, LLOC
  - CMU/SEI チェックリスト
- 作成方法、出所
  - 25%を超える修正は新規実装と同程度の工数 [Linda09]
- 言語の違い
  - ビジュアル言語、オブジェクト指向: 手続き型ほど役立たない
  - 言語補正係数表によるファンクションポイントとの関係 [LOC/FP](例: アセンブリ 320, Java 53, C 128, 表計算 6)
- コメント比率 = コメント行数 / LOC
  - 12.6%を超えると、コメントなしコードの2倍欠陥潜在率 [阿萬12]



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			
3	以前のバージョン、ビルド、またはリリース		✓	
4	商用ソフトウエア (COTS)、ライブラリを除く		✓	
5	官給ソフトウエア、再利用ライブラリを除く		✓	
6	他製品		✓	
7	ベンダー提供の言語サポートライブラリ (修正なし)			✓
8	ベンダー提供の OS またはユーティリティ (修正なし)			✓

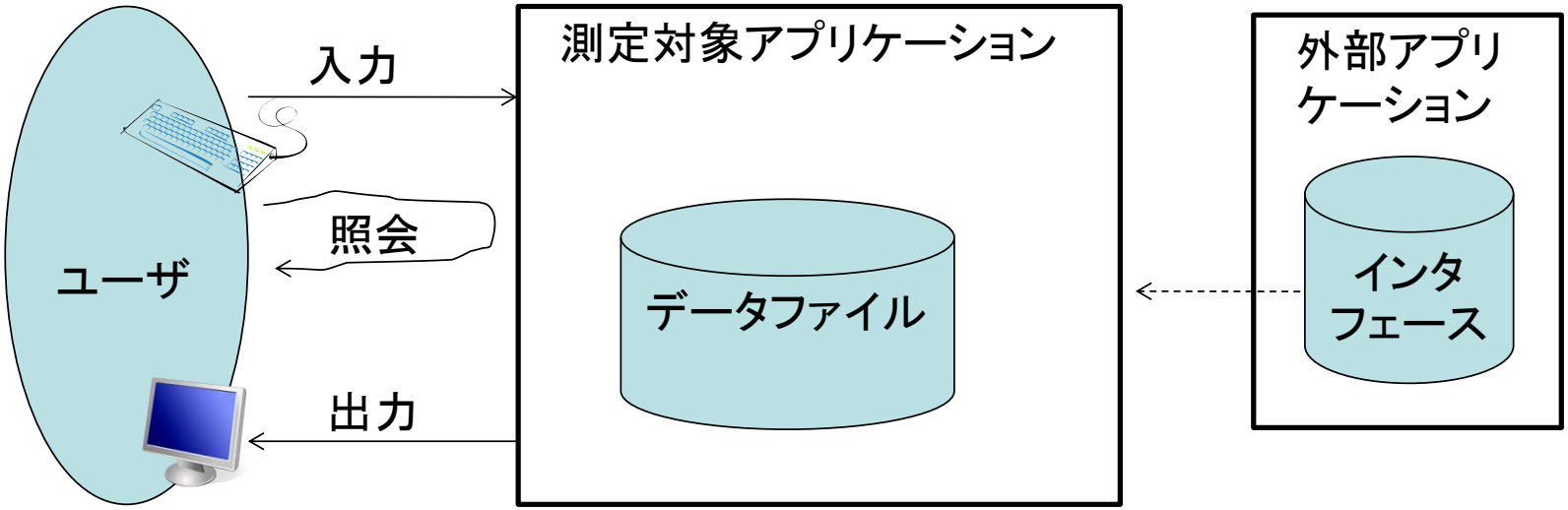
[Linda09] リンダ・M・ライルド, M・キャロル・ブレナン著, 野中誠, 鷲崎弘宜 訳 , "演習で学ぶソフトウエアメトリクスの基礎 ソフトウエアの測定と見積もりの正しい作法", 日経BP社 , 2009.

CMU/SEI-92-TR-022. ESC-TR-92-022. Software Quality Measurement: Framework for Counting Problems and Defects. http://www.sei.cmu.edu/reports/92tr020.pdf

阿萬, “オープンソースソフトウェアにおけるコメント記述およびコメントアウトと フォールト潜在との関係に関する定量分析”, 情報処理学会論文誌, 53(2), 2012.

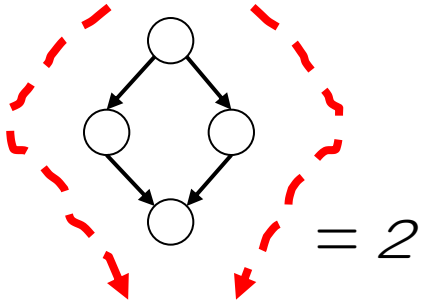
規模: ファンクションポイント法

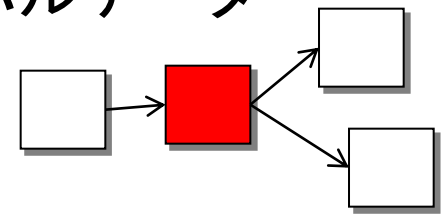
- 内部の処理や解決策ではなく問題の扱い
  - システムの入出力、インタフェース、データファイルといった「代理」（プロキシ）の数え上げ
  - 複雑さや環境などの難易度係数で調整
- 機能規模測定（Functional Size Measurement）
  - ファンクションポイント法（FP法）: 1979 オールブレクト
  - IFPUG（International Function Point Users Group http://www.ifpug.org ）結成、標準化
  - 技術独立・初期段階で利用可 ⇔ 経験の必要性、主観性・属人性



複雑さ: サイクロマティック複雑度など

- 欠陥、保守やテストの困難さの検討
  - レビュー・テスト必要箇所の特定、工数見積もり
- 関数内: サイクロマティック複雑度
  - 20が欠陥含まれやすさ閾値
  - [鷲崎10] 1 推奨、2～9 許容
- モジュール内: 凝集度
  - 単一機能 ← データ中心 ← 機能中心 ← 無関係
  - LCOM（Lack of Cohesion in Methods）< 2～5
    [Chidamber94]: 参照変数が共通しないメソッド組み合わせ数 － 共通ありの組み合わせ数
- モジュール間: 結合度
  - データ渡し ← 制御フラグ渡し ← グローバルデータ
  - ファンアウト≦ 7±2 [SESSAME]

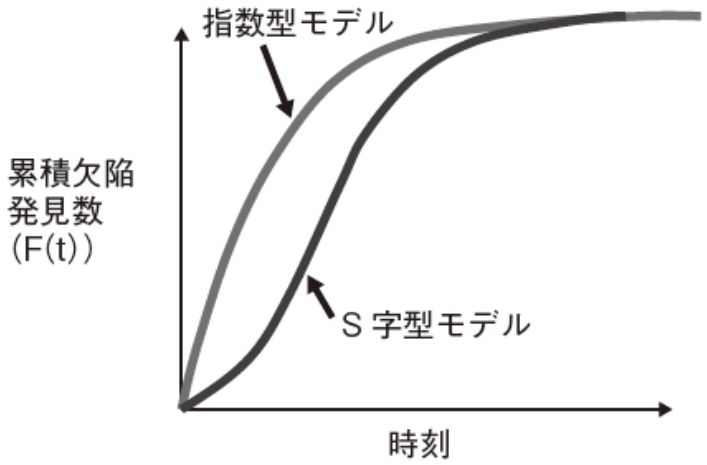




[SESSAME07] SESSAME WG2, 組込みソフトウェア開発のためのリバースモデリング, 翔泳社, 2007.
[Chidamber94] S.R.Chidamber and C.F.Kemerer, “A Metrics Suite for Object-Oriented Design”, IEEE Trans. Software Eng., vol. 20, no. 6, pp.476-493, June 1994
[鷲崎10]鷲崎弘宜,田邉浩之, 小池利和,”ソースコード解析による品質評価の仕組み“, 日経エレクトロニクス, 2010年1月25日号

欠陥: DD, DRE

- 欠陥数、欠陥発見・摘出数
  - 実測、動的・静的予測
  - 時間あたりの場合は単位に注意
- 欠陥密度（Defect Density: DD）
  - = 欠陥数 / 新規・変更LOC
  - 品質把握、プロセス全体のパフォーマンス把握
  - ドメインや稼働時間の相違の考慮が必要
- 欠陥除去率（Defect Removal Efficiency: DRE）
  - = 当該工程欠陥除去数 / 当該工程で存在した欠陥数
  - 各欠陥摘出工程の欠陥摘出能力把握、改善
  - 各欠陥の作り込み・原因工程の記録必要あり



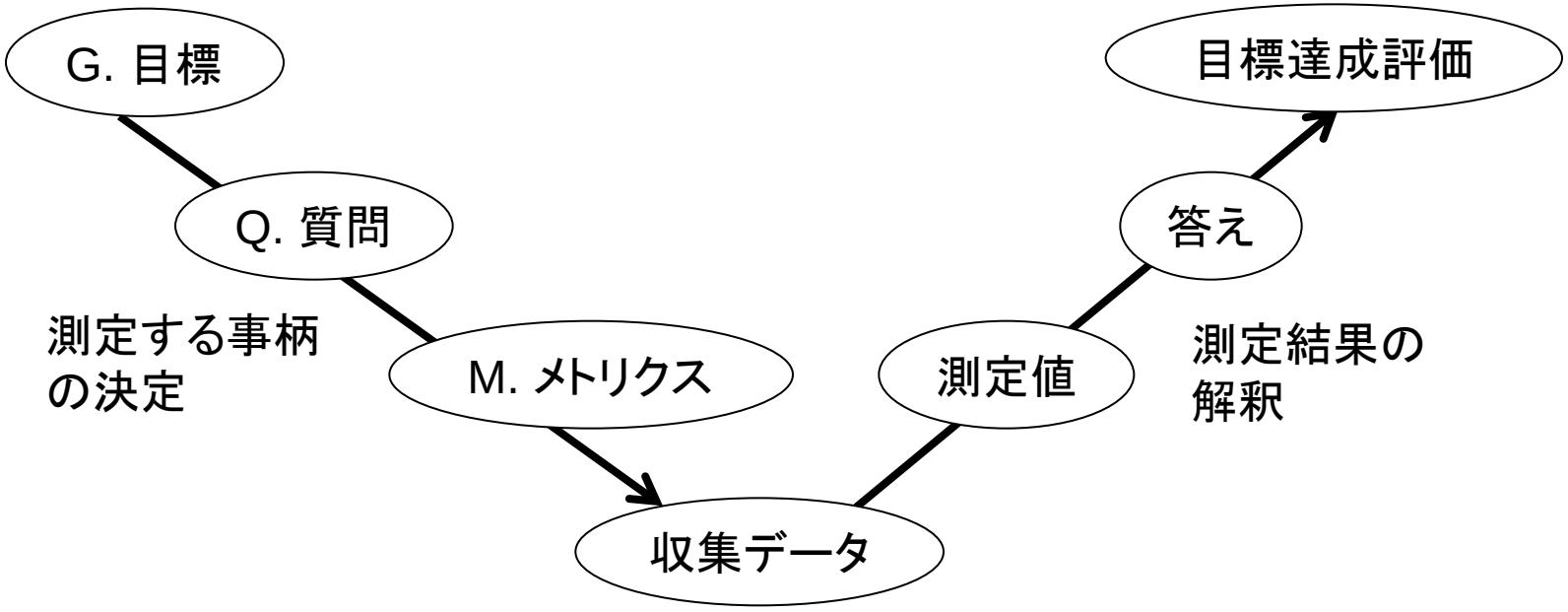
[Linda09] リンダ・M・ライルド, M・キャロル・ブレナン著, 野中誠, 鷲崎弘宜 訳 , "演習で学ぶソフトウエアメトリクスの基礎 ソフトウエアの測定と見積もりの正しい作法", 日経BP社 , 2009.

目次

- メトリクスの基本概念
- 主要なメトリクス: 規模，複雑さ，欠陥
- **GQMによるゴール指向な測定**
- GQMを用いた組込みソフトウェア品質改善事例
- まとめ

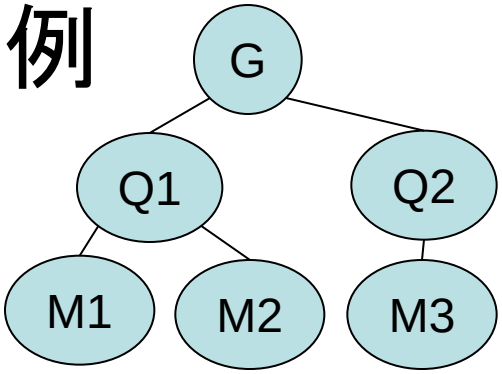
Goal-Question-Metric (GQM) パラダイム

- 明確に目標を据えて、目標に対して必要なメトリクスを対応付けるゴール指向（目標指向）な枠組み
- 目標（Goal）: 測定上の目標
- 質問（Question）: 目標の達成を評価するための質問
- メトリクス（Metric）: 質問に回答するために必要な定量的データを得るための主観的・客観的メトリクス



R. van Solingen, E. Berghout, “The Goal/Question/Metric Method” McGraw-Hill Education, 1999; ISBN 0-07709-553-7.

GQMモデル(グラフ)の例



- 目標
  - G. コードクローン含有の観点から保守性の低いコードを特定できている。
- 質問
  - Q1: コードクローンは存在するか？ → M1, M2
  - Q2: 発見されたコードクローンは保守上有害か？ → M3
- メトリクス
  - M1 : ソースコードの全体的な重複度
  - M2: モジュール単位のコードクローン含有率
  - M3: コードクローンに対するマネージャの主観的評価

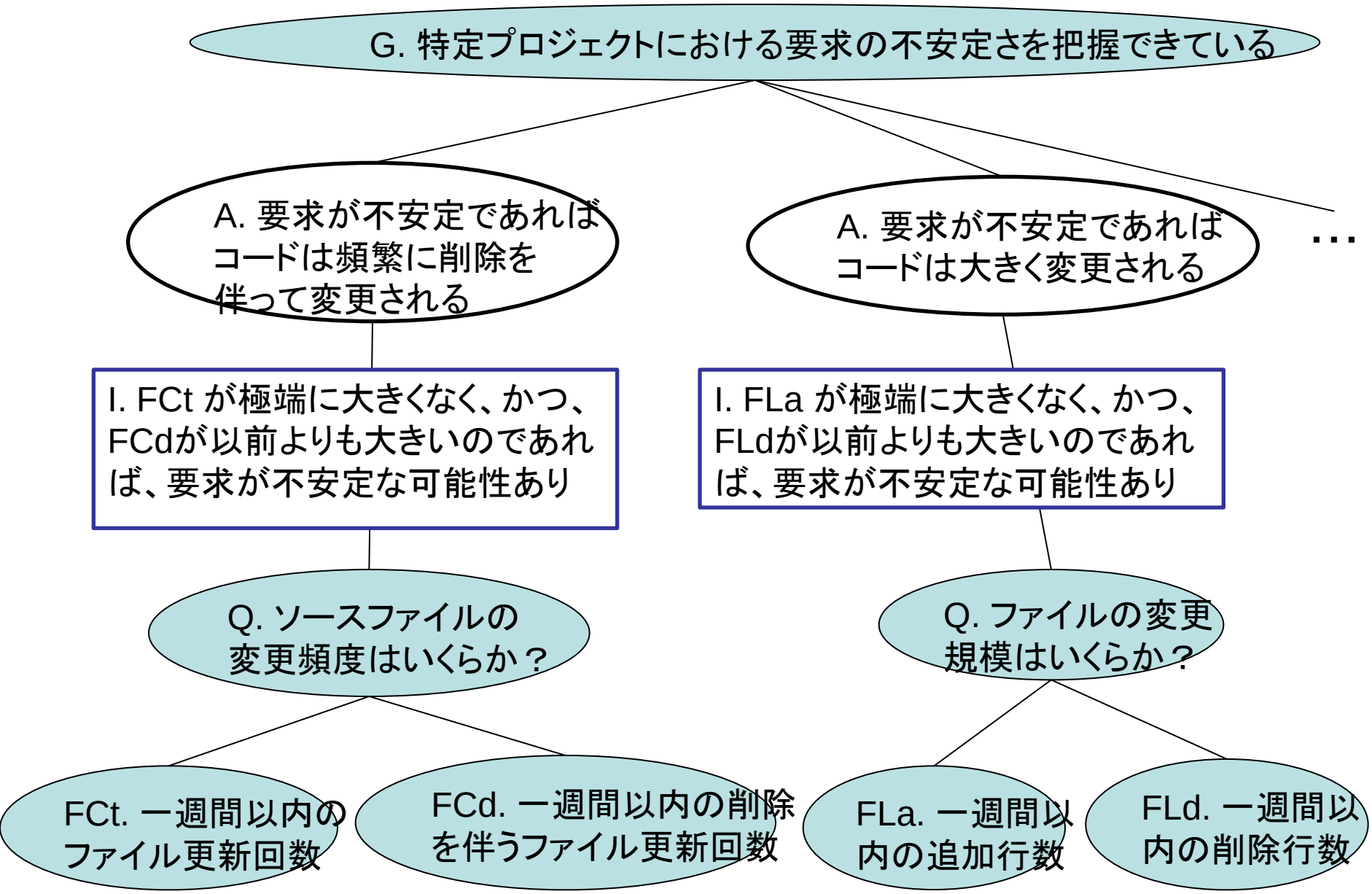
リンダ・M・ライルド, M・キャロル・ブレナン著, 野中誠, 鷲崎弘宜 訳 , "演習で学ぶソフトウエアメトリクスの", 日経BP社 , 2009.
V. Basili, et al.: Goal, Question, Metric Paradigm, Encycloperia of Software Engineering, Vol. 1(1994)
楠本真二, 肥後芳樹, “GQMパラダイムを用いたソフトウェアメトリクスの活用”, コンピュータソフトウェア, 2012.

GQMの拡張

- 上位／ビジネスゴール（参考: GQM+Strategies）
  - 測定上のGoalの上位ゴール
- 仮定（Hypothesis, Assumption）
  - Question導出にあたり仮定している事柄
  - A. 要求が不安定であれば設計は頻繁に変更される。
- 解釈（Interpretation）
  - もし「条件式」ならば「Goalはこのように達成・未達成」
  - I. もしLOCが少なすぎず、かつ、設計変更回数が一定以上ならば、要求が不安定な可能性がある。
- メカニズム（Mechanism）
  - データの収集と報告を実施する責任者、収集・報告頻度、必要なインフラ
  - 例: マネージャが毎週レポートを参照して・・・

リンダ・M・ライルド, M・キャロル・ブレナン著, 野中誠, 鷲崎弘宜 訳 , "演習で学ぶソフトウエアメトリクスの基礎 ソフトウエアの測定と見積もりの正しい作法", 日経BP社 , 2009.
Monden, Basili, et al.: Customizing GQM Models for Software Project Monitoring, IEICE Trans., 2012.

仮説や解釈を伴う例

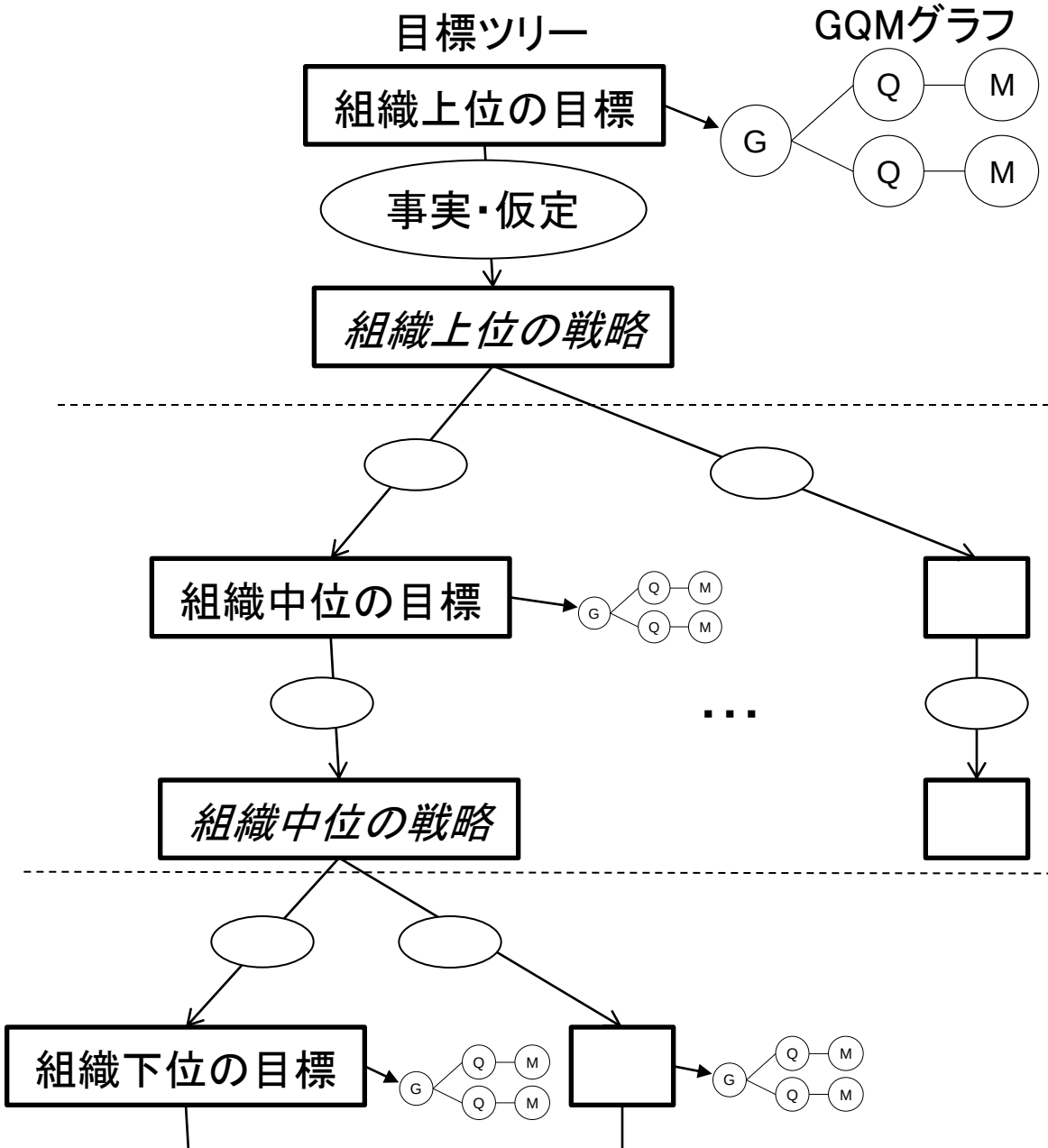


Monden, Basili, et al.: Customizing GQM Models for Software Project Monitoring, IEICE Trans., 2012.

GQM+Strategies: 組織目標・戦略・測定上の目標

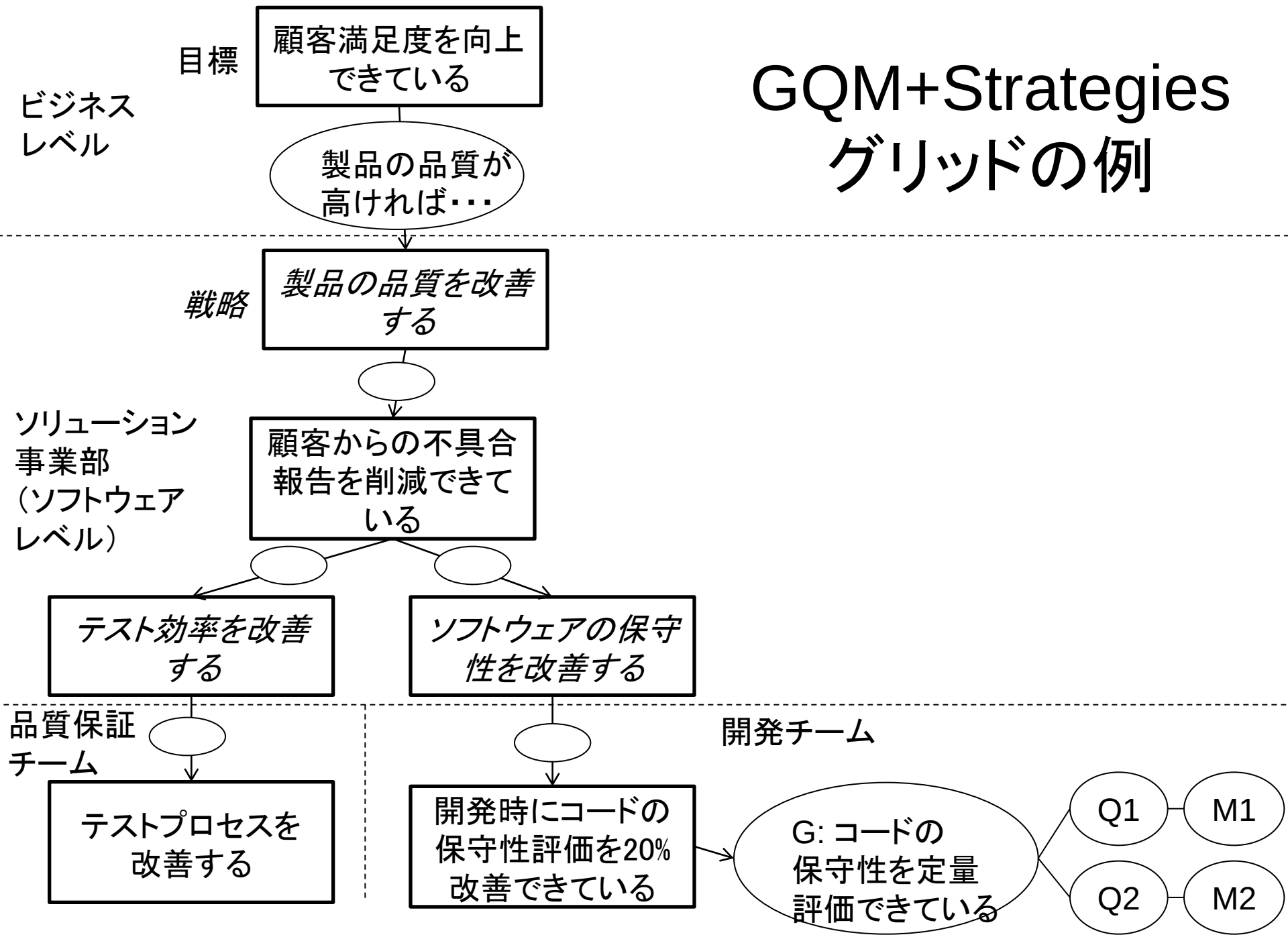
- 事実（Context）や仮定（Assumption）を明示して戦略結びつけ
- 組織上位から下位まで連鎖
- 測定上の目標を掲げたGQMグラフを紐づけ定量評価

- IESEによる開発、ツールあり
- 早稲田大学ゴール指向経営研究会、ITC 研究会 開始（’13～）



Basili, V.R., et al. “Linking Software Development and Business Strategy Through Measurement,” IEEE Computer, Vol.43, No.4, pp.57-65, 2010.
新谷勝利、平林大典: “企業・組織の目標達成とIT導入計画の整合化を実現するための手法推進”、SEC Journal、2012.

GQM+Strategies グリッドの例

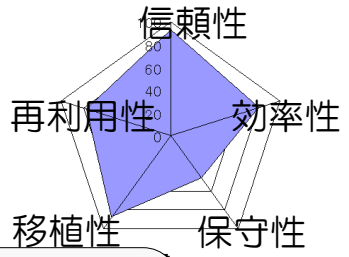


鷲崎弘宜、”ゴール指向の測定評価と留意 – GQMパラダイムと拡張 -”、メトリクス公団、Vol.1、TEF東海メトリクス勉強会、2013.

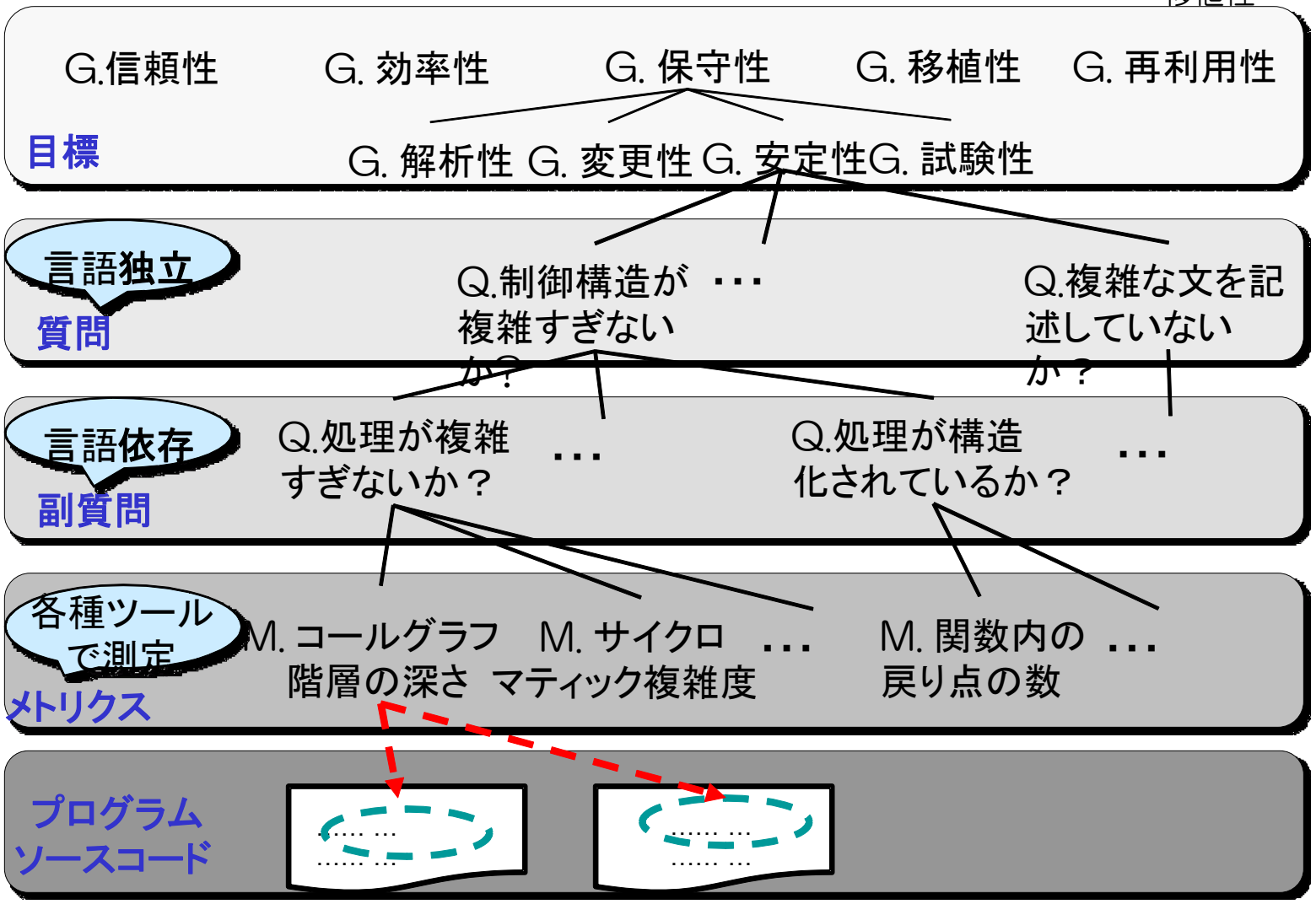
目次

- メトリクスの基本概念
- 主要なメトリクス: 規模，複雑さ，欠陥
- GQMによるゴール指向な測定
- **GQMを用いた組込みソフトウェア品質改善事例**
- まとめ

規模や複雑さから品質診断へ



- GQMパラダイムによる段階的マッピング



[Adqua] http://www.ogis-ri.co.jp/product/b-08-000001A6.html

鷲崎弘宜,田邉浩之,小池利和,ソースコード解析による品質評価の仕組み,日経エレクトロニクス 2010/1/25

対象例: ETロボコン・コード改善

- 規模: 1481 ELOC, 18クラス, 121関数
- サイクロマティック複雑度の大きい関数をリストアップ
- しかし「数」だけでは問題は分からない → 対象の詳細調査へ

品質副特性

品質特性

保守性	得点	78.6
	偏差値	36.1

解析性	得点	79.3
	偏差値	38.7
変更性	得点	89.3
	偏差値	46.2
安定性	得点	84.3
	偏差値	50.0
試験性	得点	61.5
	偏差値	35.8

ファイル名	関数名	サイクロマティック複雑度	関数の物理行数
Magi.cpp	decideSynthetic()	12	55
Tactics.cpp	getDriver(DriveStatus::eDifficultKind)	12	51
StepDriver.cpp	drive(bool)	11	119
SeesawDriver.cpp	drive(bool)	8	100
GarageDriver.cpp	driveIn(void)	8	73
	askDrive)	7	91
	Status()	7	37
	ocation()	6	81
	(void)	5	15

Question	得点	偏差値	メトリック	得点
処理が複雑すぎないか	44.5	37.7	制御フローグラフの最大ネスト数	40.9
			サイクロマティック複雑度	46.8
処理が構造化されているか	99.5	46.5	break等のないswitch文の数	91.0
要素間の結合度は低くなっているか	75.9	48.1	使用する他クラスの種類数	69.3
			自クラスを使用する他クラスの種類数	70.4
要素の規模は適切か	59.1	43.7	関数の実行コード行数	34.5
			クラスの実行コード行数	35.6
			ファイル内の関数の数	85.6
			クラス内のメソッド数	78.1

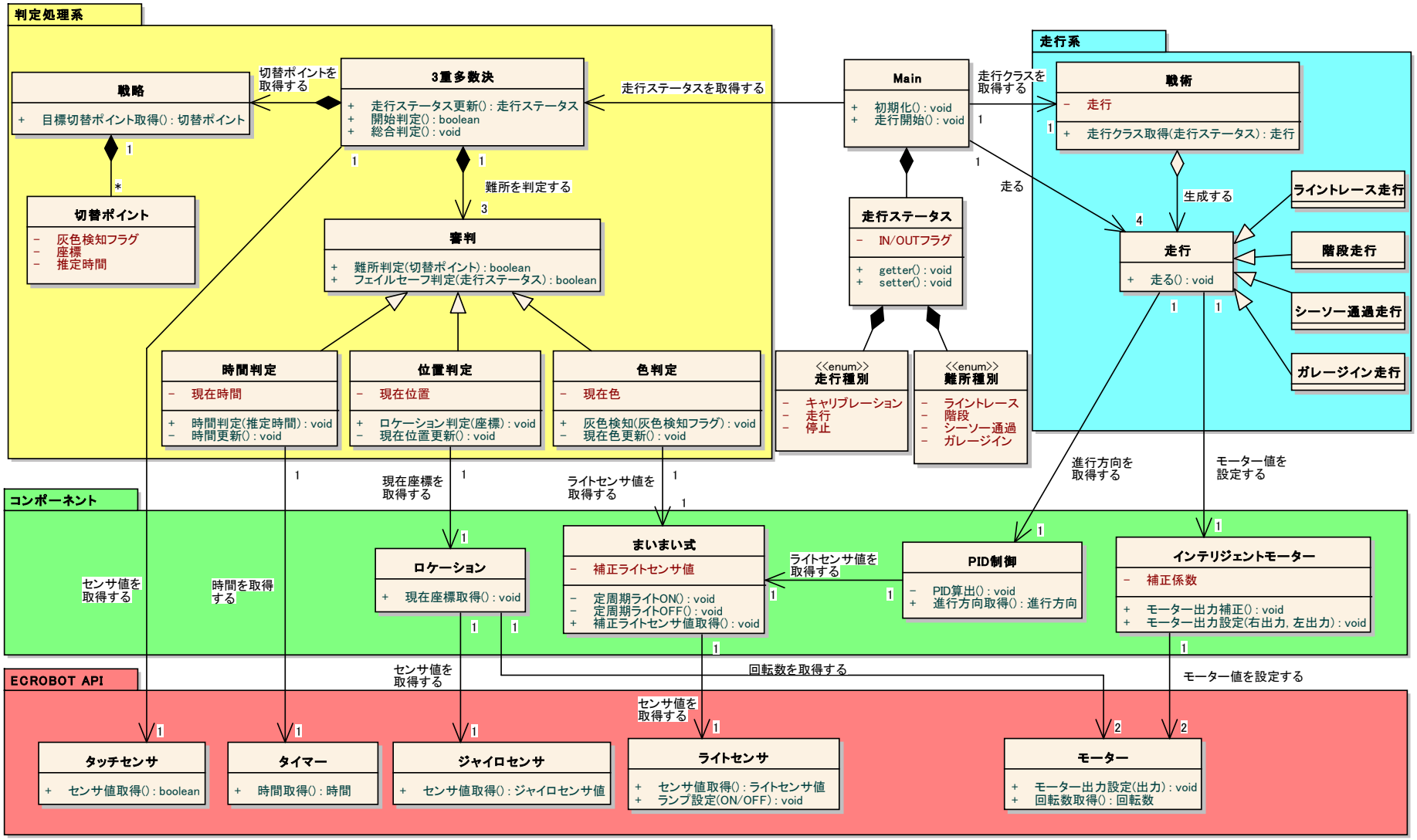
鷲崎, 阿佐美, 田邉, ETロボコンの事例で学ぶモデル活用の効能: 第2回参加チームの事例,日経エレクトロニクス, 2011年9月5日

```
//=============================================================================
// 難所走行状態の更新
void Magi::decideSynthetic()
{
    // 次に目指す難所を取得
    ChangePoint p = mStrategy.getChangePoint();

    // 事前に登録した判定クラスから難所を取得
    int decideCount = 0;
    for (SINT i = 0; i < mNumOfJudge; i++) {
        if (mJudgeAddressTable[i]->decideHard(p)) {
            decideCount++;
        }
    }

    if (decideCount >= 1) {
        switch (p.getPointKind()) {
            case ChangePoint::SEESAW_IN:
                mDifficultKindBuf = DriveStatus::SEESAW;
                mCangeCount = 0;
                break;
            case ChangePoint::SEESAW_OUT:
                mDifficultKindBuf = DriveStatus::LINETRACE;
                mCangeCount = 0;
                break;
            case ChangePoint::STEPS_IN:
                mDifficultKindBuf = DriveStatus::STEPS;
                mCangeCount = 0;
                break;
            case ChangePoint::STEPS_OUT:
                mDifficultKindBuf = DriveStatus::LINETRACE;
                mCangeCount = 0;
                break;
            case ChangePoint::OUTCOURSE_GARAGE:
                mDifficultKindBuf = DriveStatus::OUTCOURSE_GARAGE;
                mCangeCount = 0;
                break;
            case ChangePoint::INCOURSE_GARAGE:
                mDifficultKindBuf = DriveStatus::INCOURSE_GARAGE;
                mCangeCount = 0;
                break;
            default:
                break;
        }
        // 目指す難所を更新する
        mStrategy.next();
    }

    // 難所判定後、時間差で走行を切替えるケースに備える
    if (mCangeCount > 0) {
        mCangeCount--;
    } else if (mCangeCount == 0) {
        mCangeCount--;
        // 走行切替
        mStatus.setDifficultKind(mDifficultKindBuf);
    }
}
```

```
//=============================================================================
// 走行ステータスに応じて走行クラスを返却
Driver* Tactics::getDriver(DriveStatus::eDifficultKind kind)
{
    Driver* drive;
    drive = 0;
    switch(kind)
    {
        case DriveStatus::LINETRACE:
            if(mLineTraceDriver == 0)
            {
                mLineTraceDriver = new LineTraceDriver(mMaimai, mPid);
            }
            drive = mLineTraceDriver;
            break;
        case DriveStatus::SEESAW:
            if(mSeesawDriver == 0)
            {
                mSeesawDriver = new SeesawDriver(mrGyro, mMaimai, mPid);
            }
            drive = mSeesawDriver;
            break;
        case DriveStatus::STEPS:
            if(mStepDriver == 0)
            {
                mStepDriver = new StepDriver(mrGyro, mMaimai, mPid);
            }
            drive = mStepDriver;
            break;
        case DriveStatus::OUTCOURSE_GARAGE:
            if(mGarageDriver == 0)
            {
                mGarageDriver = new GarageDriver(mPositionDecision);
            }
            drive = mGarageDriver;
            break;
        case DriveStatus::INCOURSE_GARAGE:
            if(mGarageDriver == 0)
            {
                mGarageDriver = new GarageDriver(mPositionDecision);
            }
            drive = mGarageDriver;
            break;
        default:
            if(mLineTraceDriver == 0)
            {
                mLineTraceDriver = new LineTraceDriver(mMaimai, mPid);
            }
            drive = mLineTraceDriver;
            break;
    }
    return drive;
}
```

「数」から問題・原因の特定へ

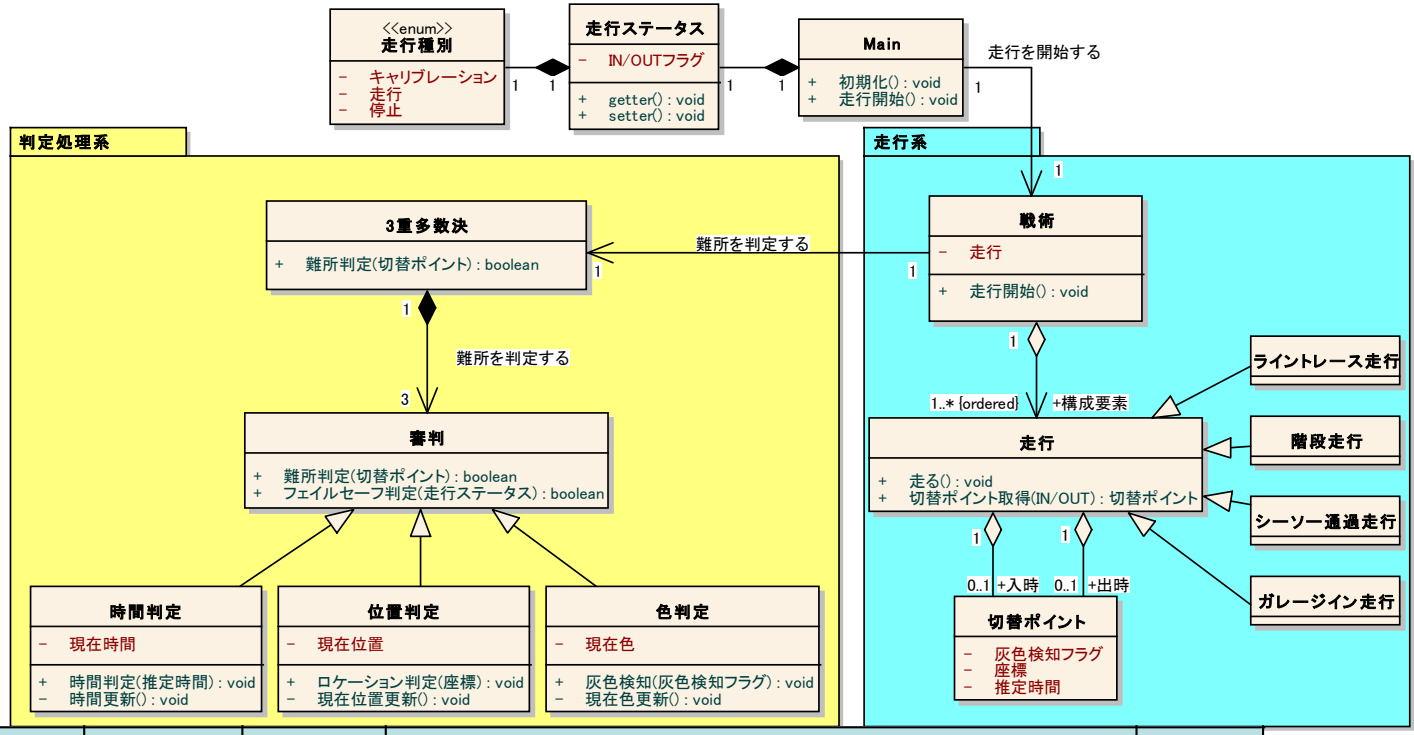
- 散らばり or もつれあい
- 「難所に関する処理・責務（関心事）」のパッケージ横断な散らばり



鷲崎, 阿佐美, 田邉, ETロボコンの事例で学ぶモデル活用の効能: 第2回参加チームの事例,日経エレクトロニクス, 2011年9月5日

改善結果

- 関心事の局所化
- 保守性改善



Question	得点	偏差値	メトリック	得点	
処理が複雑すぎないか	52.0	42.7	制御フローグラフの最大ネスト数	48.0	7.1 UP
			サイクロマティック複雑度	55.2	8.4 UP
処理が構造化されているか	98.7	46.1	break等のないswitch文の数	90.8	
要素間の結合度は低くなっているか	75.6	47.9	使用する他クラスの種類数	68.8	
			自クラスを使用する他クラスの種類数	67.7	
要素の規模は適切か	62.7	47.4	関数の実行コード行数	37.6	3.1 UP
			クラスの実行コード行数	39.6	
			ファイル内の関数の数	87.6	
			クラス内のメソッド数	81.1	

7.5 UP

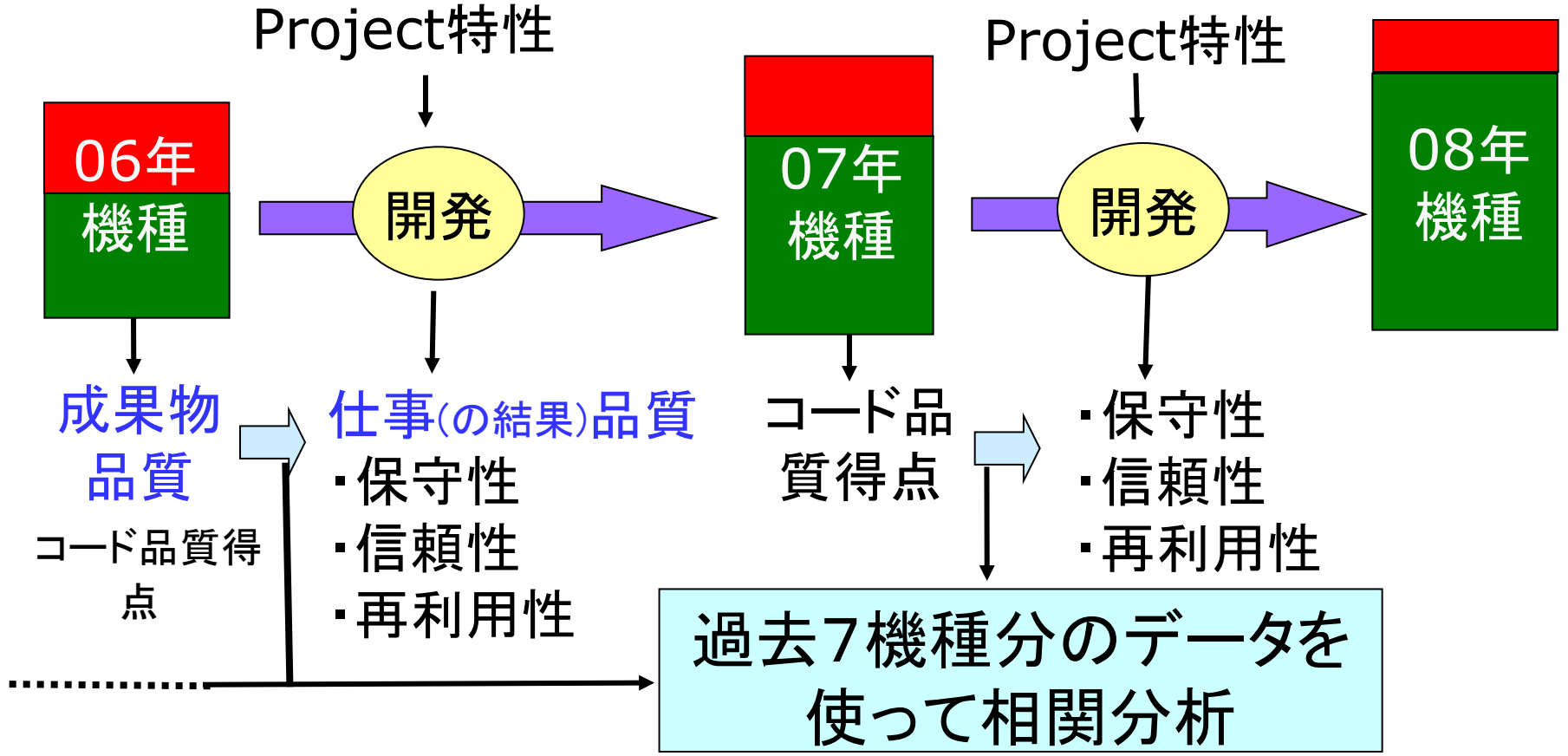
3.6 UP

鷲崎, 阿佐美, 田邉, ETロボコンの事例で学ぶモデル活用の効能: 第2回参加チームの事例,日経エレクトロニクス, 2011年9月5日

ヤマハ(株)でのコード測定事例

- 派生開発メイン、流用部分品質が多大な影響
- Adquaによるコード品質評価の利用

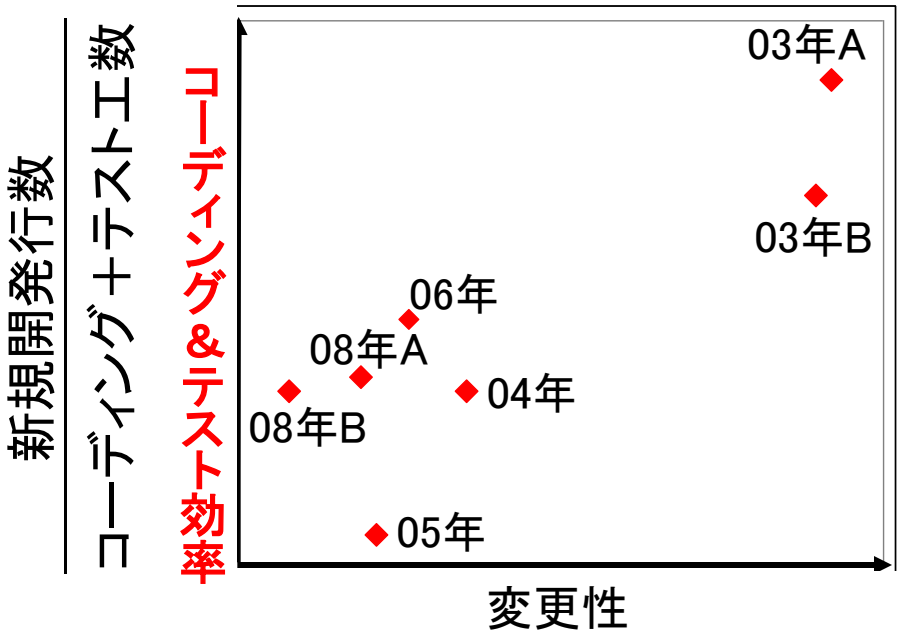
Project特性の影響を抑えるため同じチーム、プロダクトラインで

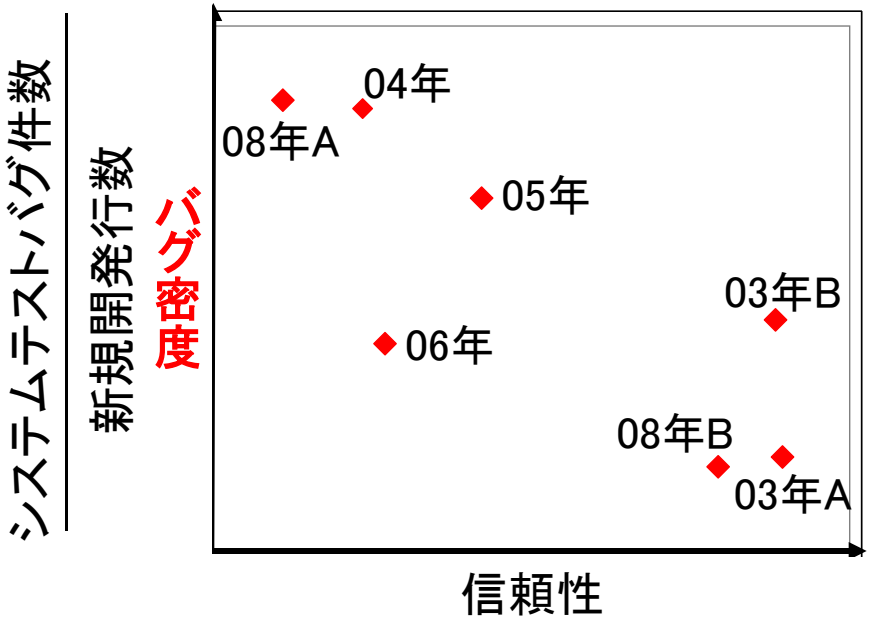


[出典] 小池利和: ソースコードの品質を直接、詳細に測る方法, SQiPシンポジウム2009

保守性・信頼性の分析結果

コード品質得点と品質指標に統計的有意な相関あり





コード品質得点の妥当性は概ね検証出来た（安心して開発者に薦められる自信が得られた）

[出典] 小池利和: ソースコードの品質を直接、詳細に測る方法, SQiPシンポジウム2009

目次

- メトリクスの基本概念
- 主要なメトリクス: 規模，複雑さ，欠陥
- GQMによるゴール指向な測定
- GQMを用いた組込みソフトウェア品質改善事例
- **まとめ**

まとめ、メッセージ

- ソフトウェア開発の厳しさとメトリクスの重要性
  - 開発の管理、品質把握と改善
- メトリクスの限界を知り意思決定に役立てる
  - 測定方法、信頼性、妥当性、負のホーソン効果
  - 基本的なメトリクス: 規模、複雑性、欠陥
- GQMによる目標指向の測定
  - 測定と改善のプロセス
  - GQM+Strategiesによる組織目標へのひもづけ

より詳しくは

- メトリクス全般
  - リンダ・M・ライルド, M・キャロル・ブレナン著, 野中誠, 鷲崎弘宜 訳, "演習で学ぶソフトウエアメトリクスの基礎 ソフトウエアの測定と見積もりの正しい作法", 日経BP社 , 2009.
- GQM
  - 鷲崎弘宜、”ゴール指向の測定評価と留意 – GQMパラダイムと拡張 -”、メトリクス公団、Vol.1、TEF東海メトリクス勉強会、2013.
  - 楠本真二、肥後芳樹: “GQMパラダイムを用いたソフトウェアメトリクスの活用”、コンピュータソフトウェア、29(3)、pp.29-38、2012.
  - Basili, V., Caldiera, C., and Rombach, D.: “Goal, Question, Metric Paradigm, Encyclopedia of Software Engineering,” Vol.1, pp. 528–532, 1994.
- GQM+Strategies
  - Basili, V.R., Lindvall, M., Regardie, M., Seaman, C., Heidrich, J., Munch, J., Rombach, D., Trendowicz, A.: “Linking Software Development and Business Strategy Through Measurement,” IEEE Computer, Vol.43, No.4, pp.57-65, 2010.
  - 新谷勝利、平林大典: “企業・組織の目標達成とIT導入計画の整合化を実現するための手法推進”、SEC Journal、2012.
- 事例
  - 鷲崎弘宜, 田邉浩之, 小池利和, “ソースコード解析による品質評価の仕組み”, 日経エレクトロニクス, 2010年1月25日号, 2010.
  - 鷲崎弘宜, 阿左美勝, 田邉浩之, “モデル活用の効能: 第1回 モデルを書く意味”～ “モデル活用の効能: 第4回 参加チームの事例3”,日経エレクトロニクス, 日経BP社, 2011年8月8日号～11月号