信頼性向上へ向けた メトリクスと統計処理の活用

愛媛大学大学院
理工学研究科 電子情報工学専攻
阿萬　裕久　(あまん・ひろひさ)
aman@cs.ehime-u.ac.jp

ソフトウェアの信頼性

- 特定の動作条件下で，特定の時間に，ソフトウェアが故障なく動作する確率のこと [1]
  - 故障 ＝ **振舞いが顧客の期待に反すること**
  - **ハードウェアの信頼性でも同様のことがいえる**

ソフトの信頼性 ＋ ハードの信頼性
⇨ システム全体の信頼性

ハードウェア設計のソフトウェア化

- **近年，**ハードウェアの設計もソフトウェアで**行われるのが主流のようである**
  - **ハードウェア記述言語** (HDL) **が使われる**

    ただし，意味論は大きく異なっているので，すぐにまとめて議論できるレベルではないが・・・

- **いずれにせよ，**システムの信頼性**において，**ソフトウェアが果たすべき役割は極めて大

ソフトウェア信頼性の注目点

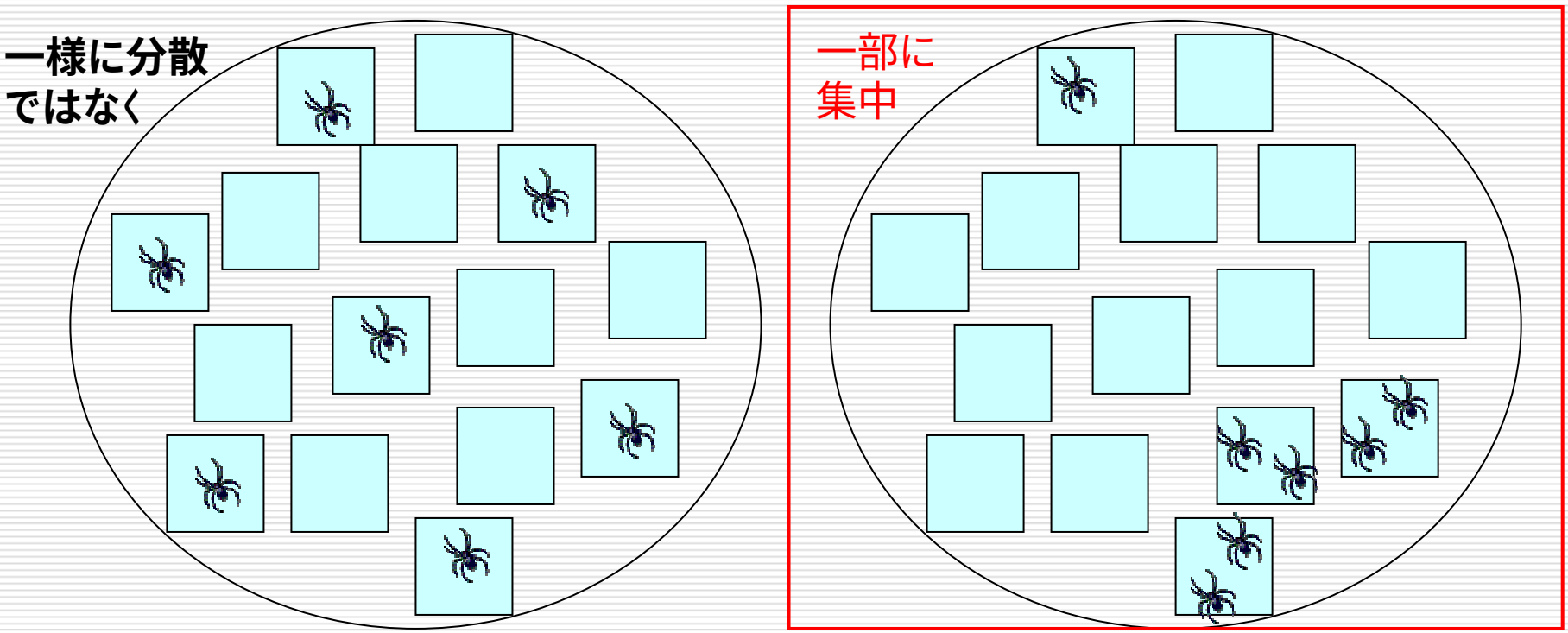
- **故障の源である欠陥（フォールト：fault）が**
  - どのように生じたか
  - どこに生じたか

- **欠陥（フォールト）の作り込みが，**
  - **物理現象（自然現象）的ではなく，**
  - **人為的に，しかし意図されずに行われる**

  ところに検出・解析の難しさ

欠陥の分布

- 欠陥の約80%は，約20%のモジュールに存在[2]



Pareto 原理 (Pareto principle)

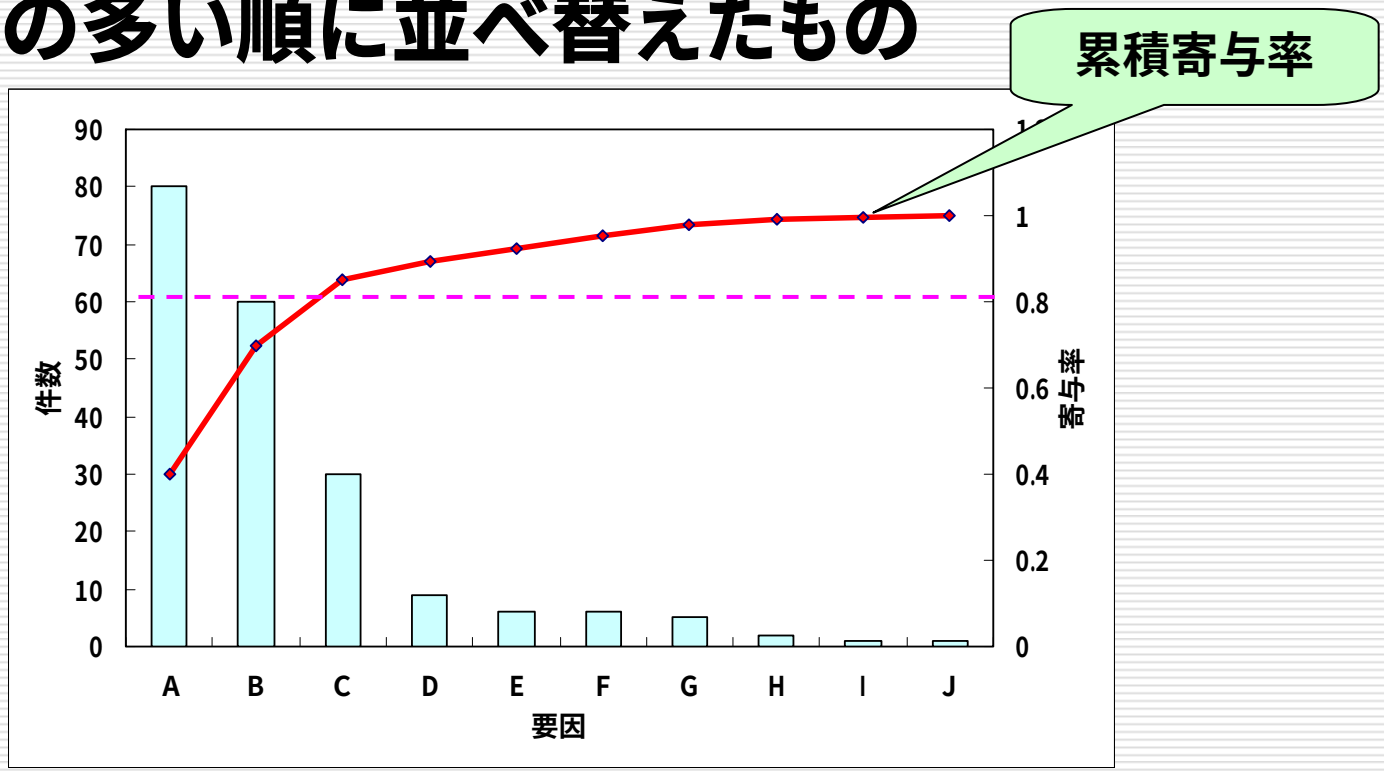
- 期待する効果の80%は (しばしば) 活動の20%から導き出される

Vilfredo Pareto (1848-1923)

- **イタリアの経済学者**
- **「収入の**80%**が人口の**20%**から得られる」**
- Pareto **の法則** (Pareto's law) **とも呼ばれる**

【参考】パレート図 (Pareto chart)

- 横軸に要因，縦軸に件数 (割合，寄与率)
- 件数の多い順に並べ替えたもの

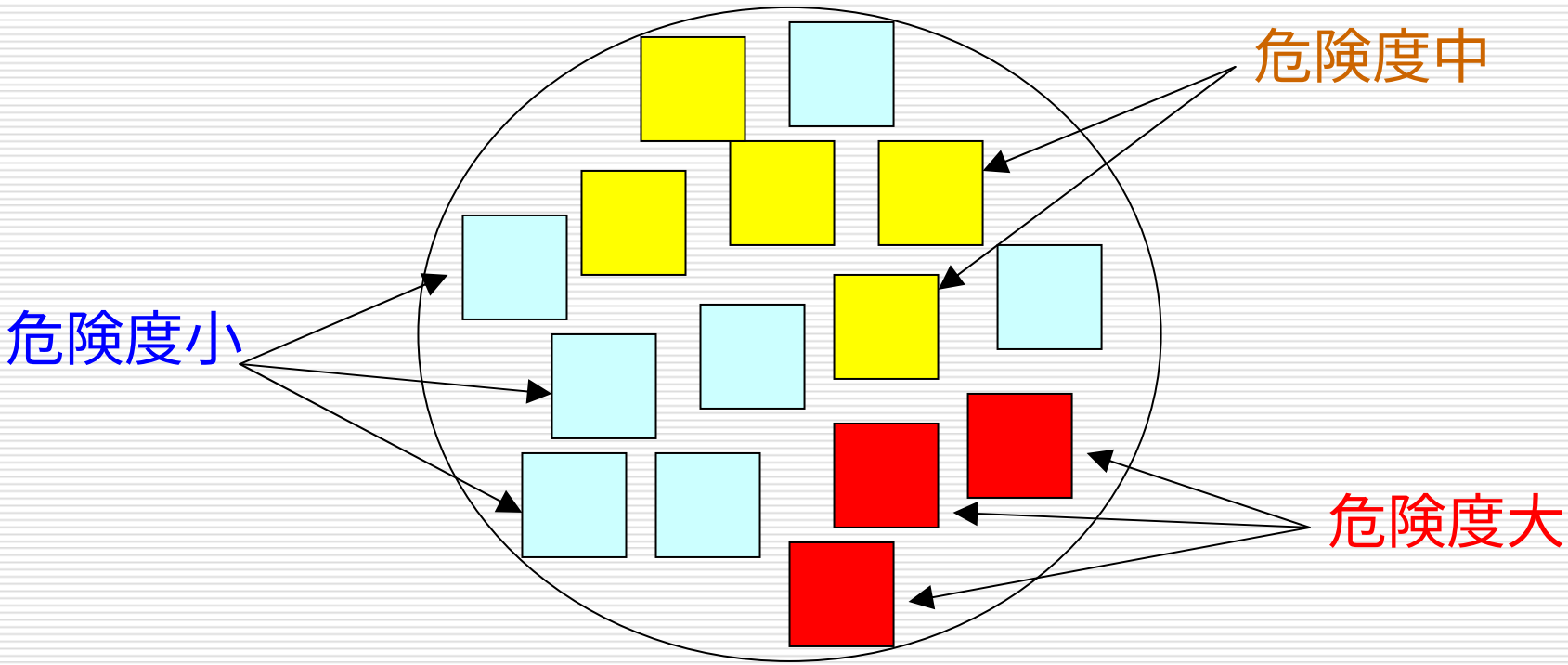


ソフトウェア工学でのパレート原理(一部)

- DOS/VS OS **における約** 500 **個の** 20% **の**モジュールでエラーの約 80% **を検出**[3](1975)
- 再作業の 80% は 20% の欠陥による[4](2000)
- **ある通信切り替えシステムにおける**エラーの 60% は 20% のモジュールに**含まれていた**[5](2000)
- **ある通信システムにおける**欠陥の 63～70% は 20% のモジュールに**含まれていた**[6](2007)

より効果的な品質・信頼性の確保

- 品質・信頼性の「怪しそうな」部分から先にチェックしていくべきと考える



基本的な考え方は血液検査

- **人間の健康管理**

サンプル血液の分析

異常値を示すと
疾病の可能性

比重
赤血球数
血色素量
網赤血球数
血小板数
赤血球沈降速度
白血球数
・・・・・・

ソフトウェアでも血液検査

- **疾病　　欠陥**

ソフトウェアの分析

コード行数
コメント文の密度
凝集性
モジュール性
構造的複雑さ
提供機能数
再利用率
・・・・・

メトリクスでの
測定値を基に検討

メトリクス　＝　良さ評価？

- メトリクスがソフトウェアの良さを評価しているとは限らない
- **解釈･対処するのは，あくまでも人間**
  - **血液検査**

    医者**による診断，精密検査，治療へ**
  - **メトリクスによる測定**

    技術者**によるレビュー，テスト，保守へ**

    ※メトリクスで異常なし　→　良質なソフトウェア？

    そうでもない．．．

現状のメトリクスの確信度は，血液検査ほど高くない

メトリクスによる測定

- **あくまでも**メトリクスの役目**は,**
  ソフトウェアの状態を数値・記号で表すこと

  - **メトリクスに求められること**
    - 役に立つ！
    - 測定しやすい
    - わかりやすい (**何を表現しているのか**)
    - 冗長でない

やっぱりシンプル**なの**
が一番かもしれない

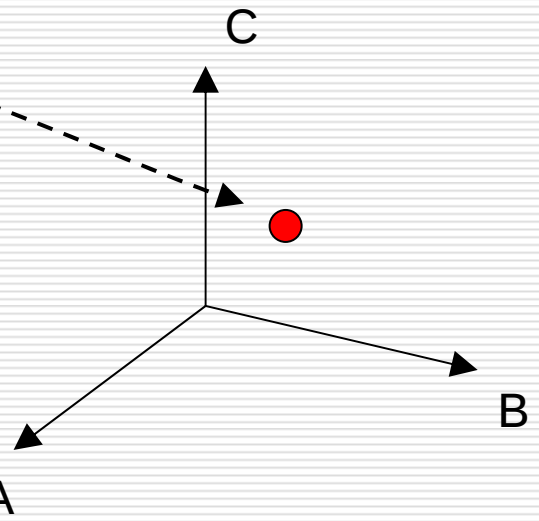
冗長でない

- ***n*** 個のメトリクスで測る**なら**
  - ***n*** 次元の情報空間を形成**すべき**

ソフトウェア

測　定

メトリクスA
メトリクスB
メトリクスC



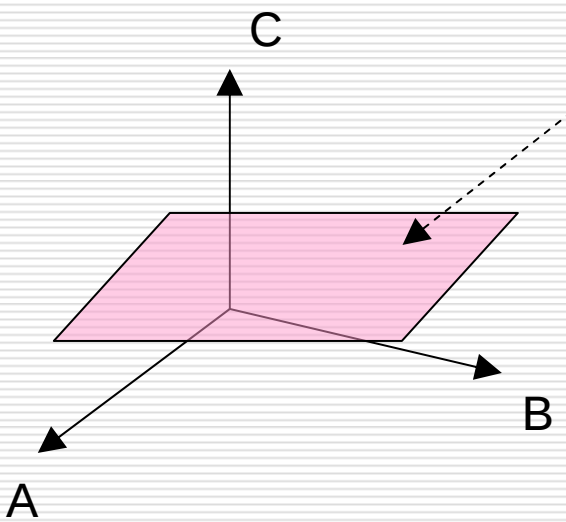
A，B，C　はそれぞれ独立した視点で測定できているか？

冗長でない： メトリクスは独立か？

- **もしも，メトリクス** A，B，C **の間に**

  1.2A - 3B = 0.5C + 10

  といった関係があったら？　実質は 2次元



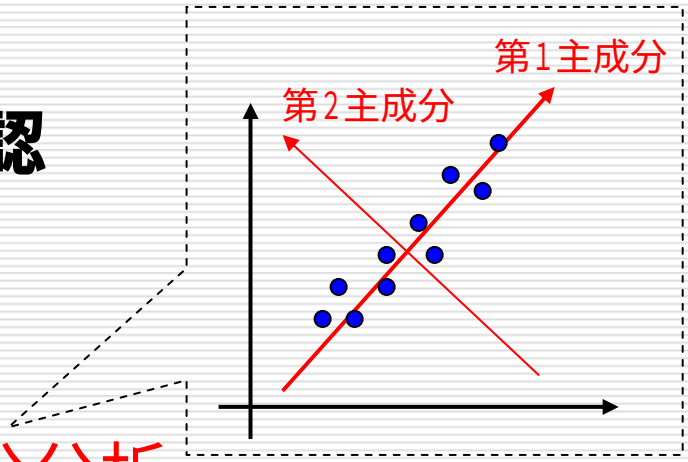
いかなるソフトウェアの測定値も，このような平面上に乗ることに

ソフトウェアの状態表現が
少数のメトリクスに影響されてしまう

一部の声の大きな人たちの意見に皆が流されてしまうようなもの

冗長性のチェック

- 定性的 (解析的) チェック
  - **メトリクス値の定義式を確認**



- 定量的チェック
  - **収集データに対する**主成分分析
    - **使用しているメトリクス集合は実質何次元か？**
  - **簡単なのは**メトリクスどうしの相関**をチェック** [7]

メトリクスの利用例（1）　1次元

- まずは1個のメトリクスで検討してみる

- シンプルさを考慮し，サイズメトリクスの一種である「文の数」で実験

  ※コード行数（LOC）の方が馴染みやすいが，
  　コーディングスタイルの影響が出てしまう

メトリクスの利用例 (1)　1次元

- **実験対象**

  オープンソースソフトウェア (記述言語 Java)
  Relaxer, JBoss, jEdit **におけるクラス** 3,514 **個**

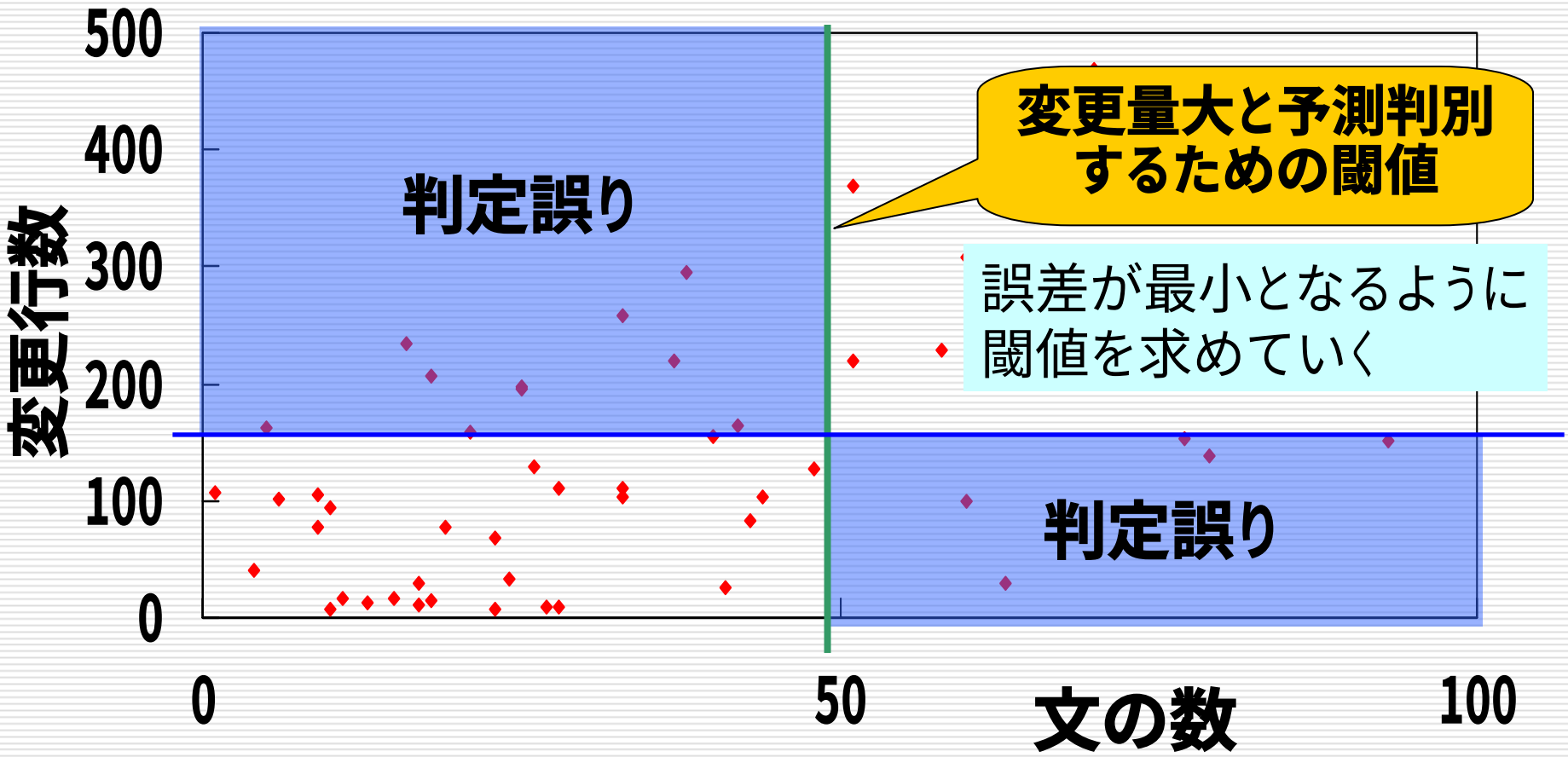
- **目的**

  バージョンアップ時の変更量が特に多いクラスを予測

  ※ここでは上位5%点, 154行以上の変更を採用

大きなコードは，それだけ変更量も多いのではないか？
そのとき，コードの規模はどれくらいが望ましいのか？

メトリクスの利用例 (1)　1次元



メトリクスの利用例 (1)　1次元

- **結果**[8]
  - 文の数 ＞ 113　**だと変更量は特に多い傾向**
    行数に換算すると, 概ね 250～400行が閾値
  - **的中率は約**77％
  - **同じ閾値を別のソフトウェアに対して適用** (408個**のクラスで検証実験**)： **結果, 的中率は約**73％

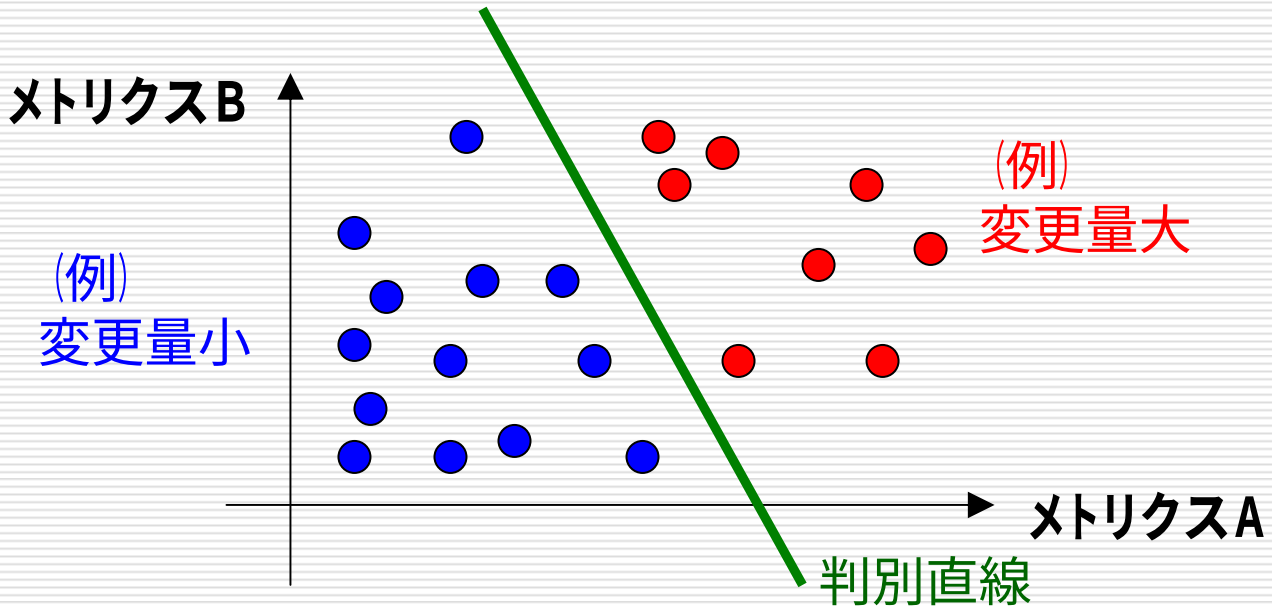
メトリクスの利用例（1）　1次元

- **シンプルな方法ではあるが，見つけたいものの約7割は予見できている**
  - **事前に検出したいクラスやモジュールを設定し，それらを混ぜて数百個単位のサンプルを用意**
  - **予測判別誤差が最小となる閾値を計算**

⇨ 開発時の品質基準に使えるのでは？

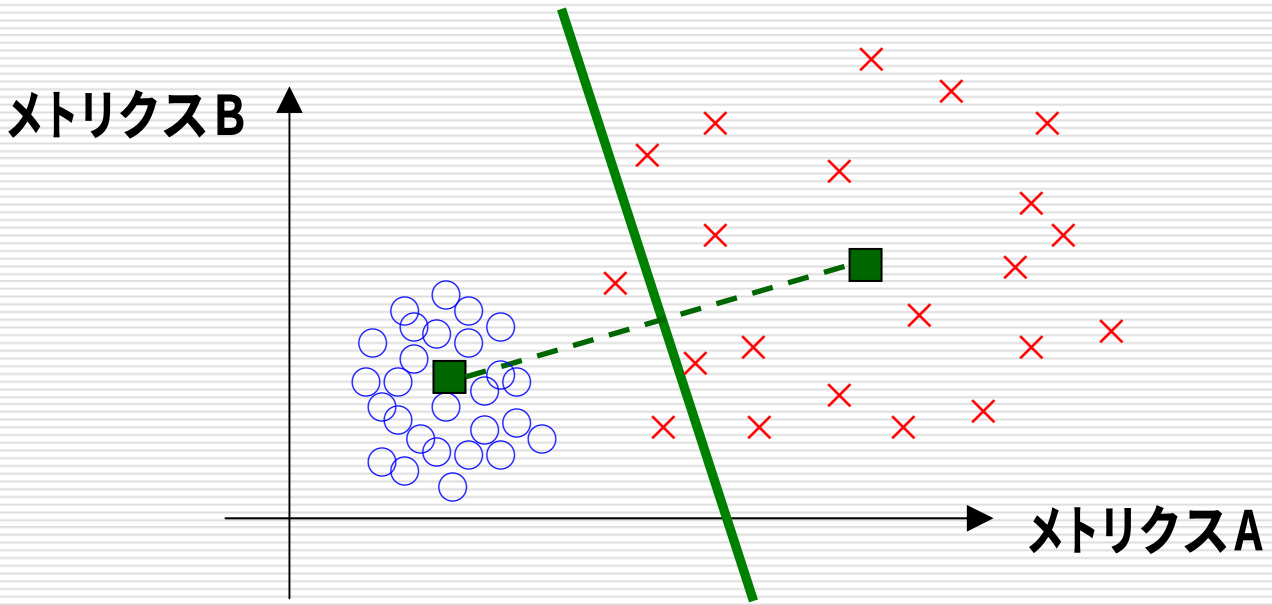
メトリクスの利用例（2） 多次元

- メトリクスを複数個用いることで，ソフトウェアが多次元空間へマッピングされる
- その中で目的のソフトウェアを判別



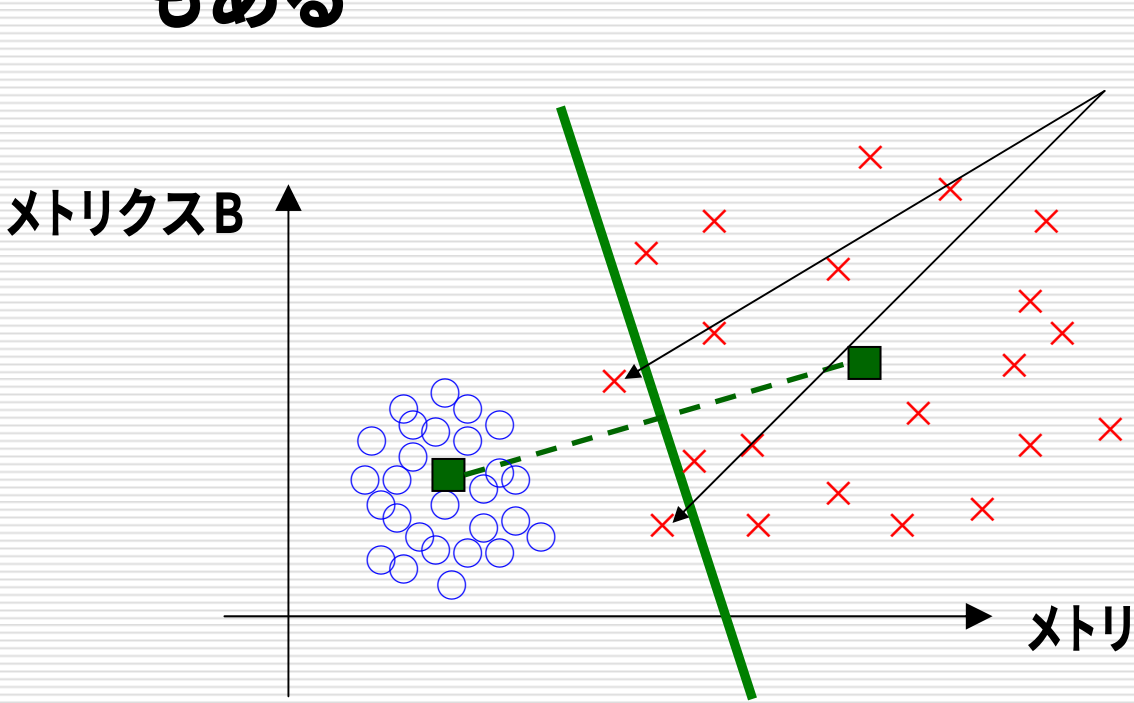
メトリクスの利用例 (2)　多次元

- パターン認識のような話になる
- 最も単純なのは，垂直二等分線の概念



メトリクスの利用例 (2)　多次元

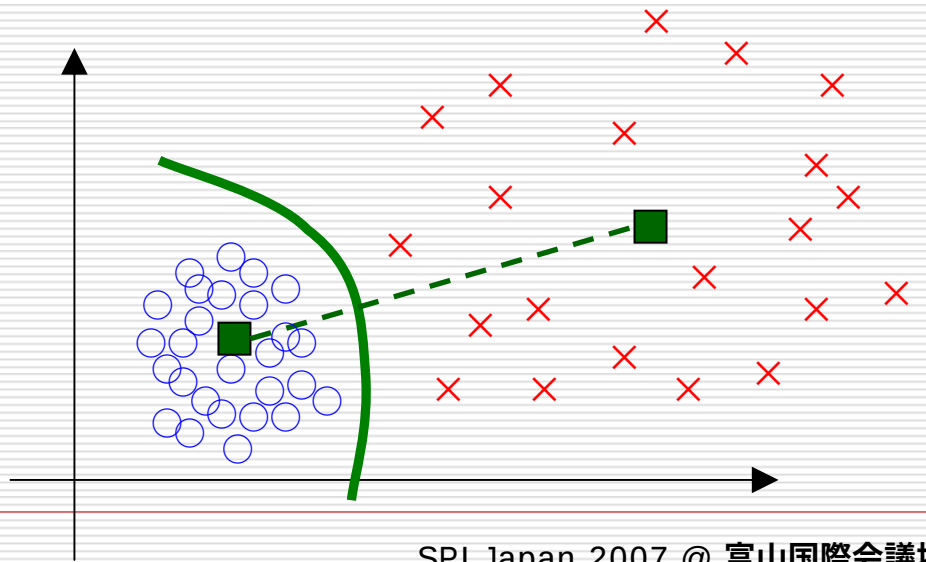
- しかし単純な距離 (ユークリッド距離) だと問題もある



データのばらつきによって，距離の価値が変わる

メトリクスの利用例（2） 多次元

- **マハラノビス**（Mahalanobis）**距離**
  - **ユークリッド距離だと** $x^T x$ **（ベクトルの内積）**
  - **マハラノビス距離だと** $x^T S^{-1} x$ **（分散・共分散行列の逆行列が入る）**



※直感的には
「ばらつき」の大きさ
で割ったようなイメージ

メトリクスの利用例 (2)　多次元

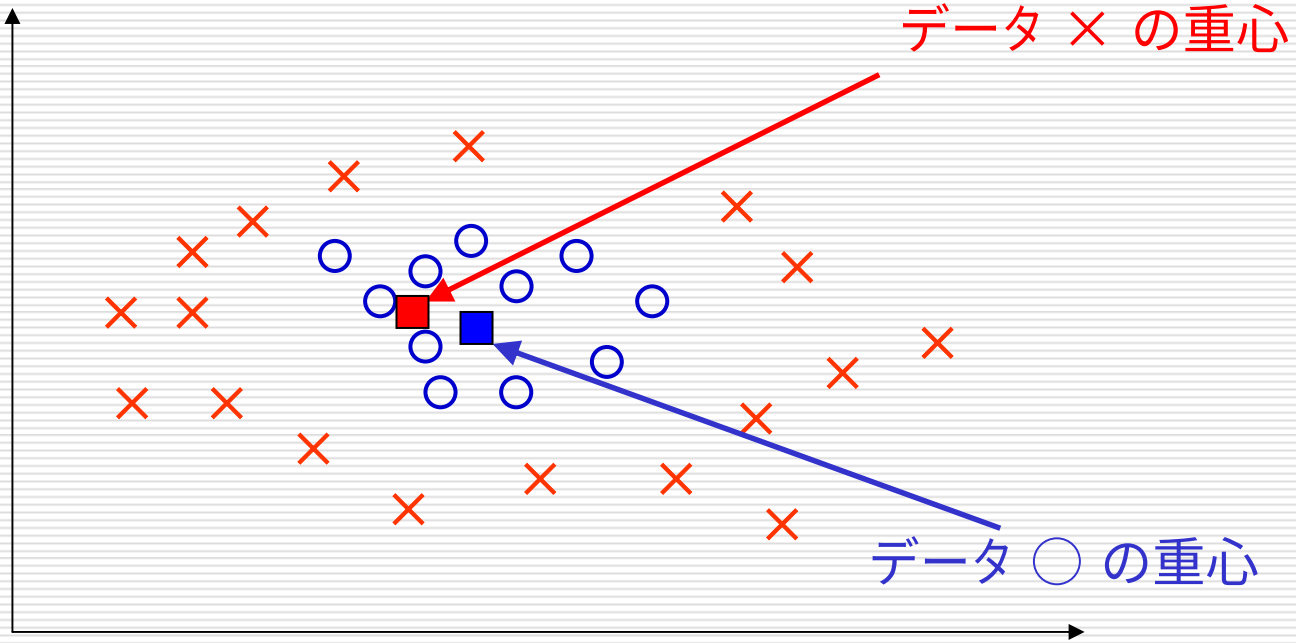
- **マハラノビス距離を使って判別実験を実行**

メトリクス	説明
文の数	クラスにおけるメソッドの宣言文と実行文の数
NCM	クラスにおけるクラス変数の数
NMA	クラスにおける新たに追加されたメソッド数
NMO	クラスにおけるオーバーライドされているメソッド数
PIM	クラスにおけるパブリックなインスタンスメソッド数

残念ながら結果は「文の数」単体とあまり変わらず．．．

メトリクスの利用例 (2)　多次元

- **2種類の標本における重心点が近いといった問題が影響 (標本空間での問題)**



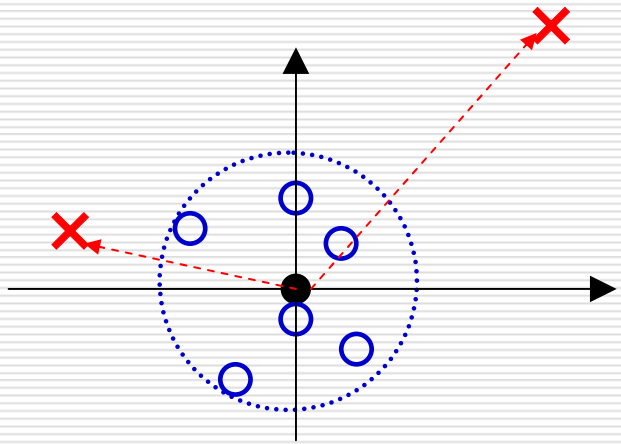
メトリクスの利用例 (3)　多次元その2

□ マハラノビス-タグチ法

あらかじめ一方のグループの
みの集合**を作り**

- **その**重心点を原点**として**
- マハラノビス距離で正規化**された**

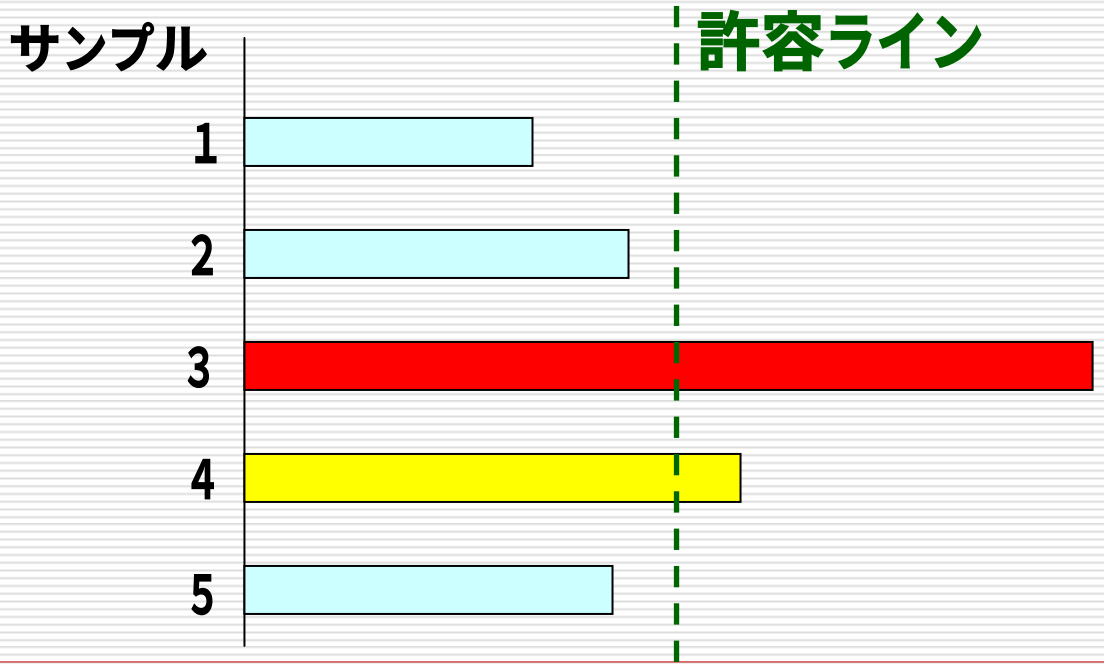
距離空間を形成する



この空間において分析対象標
本の位置**を求め，判別を行う**

メトリクスの利用例 (3)　多次元その2

- **メトリクス毎の**ばらつきは均一化**される**
- **良質サンプルからの**距離が異常度**になる**



メトリクスの利用例 (3)　多次元その2

- **実験**

  オープンソースソフトウェア Eclipse, Azureus, jEdit 5,760**個のクラスに対し，バージョンアップ時の変更量が特に大きくなると予想されるものを判別**

- **結果**：有意水準1%**で以下の仮説がいえた** [9]

  (変更量小**と予想されたものの変更量**) ×2

  ＜　(変更量大**と予想されたもの変更量**)

まとめ

- 欠陥は一部に集中**する傾向にある**
  - 早めに「あやしい」部分を**見つけましょう**
- **メトリクスは**ソフトウェアの状態を写し出す
  - シンプルなものを組み合わせて**いくと良い**
  - 冗長にならないように
- **メトリクスに**振り回されず**，測定値を**有効利用
  - データの蓄積と整理**が重要**
  - **その上で**閾値を決めていく**と基準になる**
  - **データの**分布にも注意**が必要**

参考文献

[1] **片山卓也, 土居範久, 鳥居宏次, ソフトウェア工学大事典, 朝倉書店, 東京,** 1998.

[2] Albert Endres, Dieter Rombach, A Handbook of Software and Systems Engineering, Pearson Education, 2003 (**吉舗紀子訳,** EASE**プロジェクト監修, ソフトウェア工学・システム工学ハンドブック, コンピュータ・エージ社, 東京,** 2005) .

[3] Albert Endres, “An analysis of errors and their causes in system programs,” ACM SIGPLAN Notices, vol.10, issue 6, June 1975.

参考文献

[4] B. Boehm, V.R. Basili, “Gaining intellectual control of software development,” IEEE Computer, vol.33, issue 5, pp.27-33, May 2000.

[5] N.E. Fenton, N. Ohlsson, “Quantitative Analysis of Faults and Failures in a Complex Software System,” IEEE Trans. Software Eng., vol.26, no.8, pp.797-814, Aug.2000.

[6] G. Andersson, P. Runeson, “A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems,” IEEE Trans. Software Eng., vol.33, no.5, pp.273-286, May 2007.

参考文献

[7] **阿萬裕久ほか, “主成分･相関分析によるメトリックスの定量的検証法,” 電子情報通信学会論文誌**D-I, vol.J85-D-I, no.10, pp.1000-1002, Oct.2002.

[8] H.Aman et al., “A Simple Predictive Method for Discriminating Costly Classes Using Class Size Metric, ” IEICE Trans. Inf. & Syst., vol.E88-D, no.6, pp.1284-1288, June 2005.

[9] H.Aman et al., “A Model for Detecting Cost-Prone Classes Based on Mahalanobis-Taguchi Method,” IEICE Trans. Inf. & Syst., vol.E89-D, no.4, pp.1347-1358, Apr. 2006.