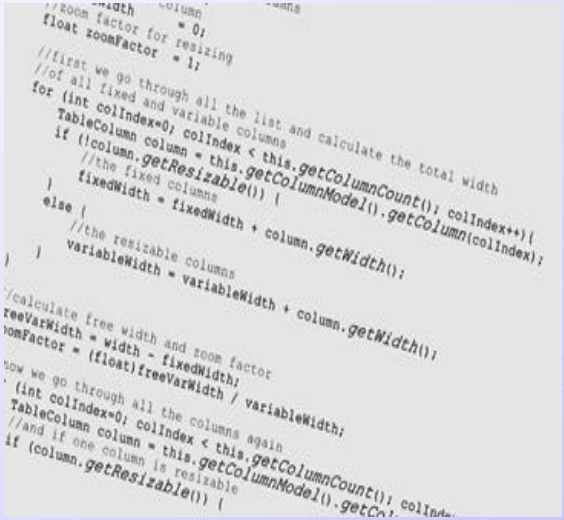


Software Quality Workshop @ NetObjectDays2004

Assessing and interpreting object-oriented software complexity with structured and independent metrics

R. Neumann
D. Klemann

Structure

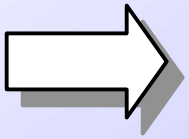
1. Introduction
2. Preparing data
3. Removing metrics correlation
4. Interpreting complexity results
5. Praxis example
6. Prospects

Why assessing complexity as a fault source

- Complexity from a cognition view

Difficulty to understand system behaviour even when all system parts are understood

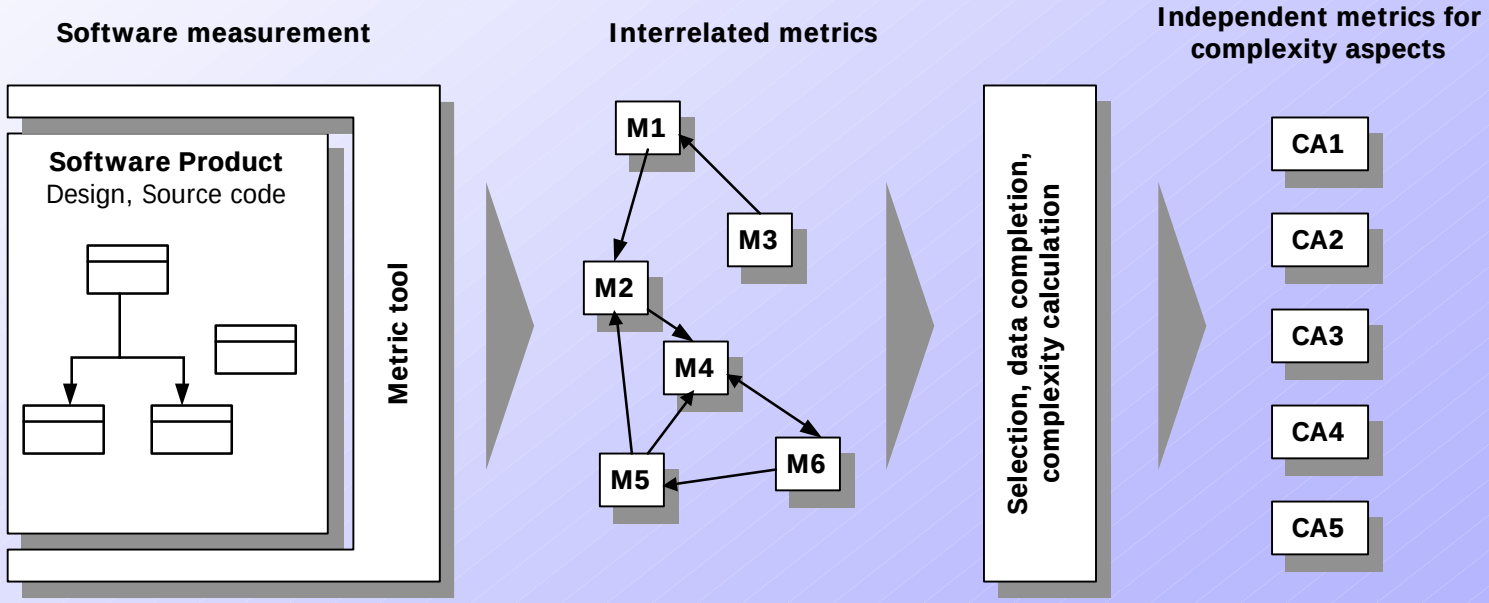
- Complicates overview on side effects
  - Cross-references from invocation and inheritance
- Overlooked side effects can cause faults



Complexity aspects need to be surveyed

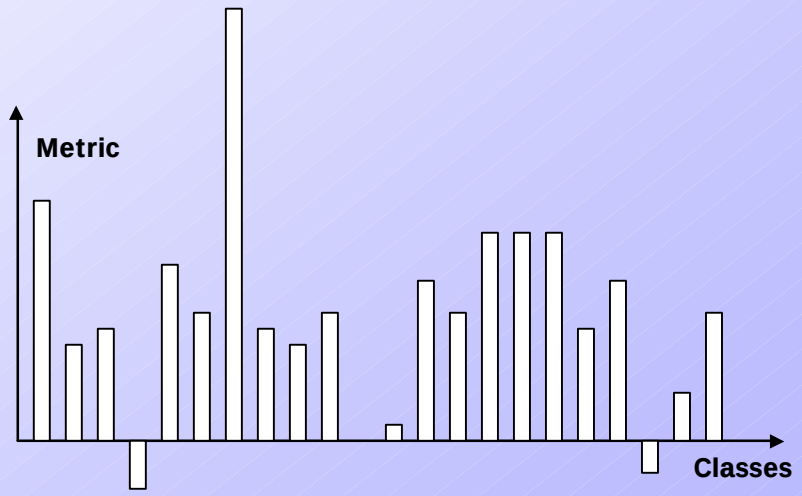
Complexity measurement process

- Metrics based software product assessment
- Inspection of raw data for missing or outlier values
- Metrics selection with a classification structure



Data Preparation

- Missing data tampers statistics (e.g. average with no values)
- Techniques for treating missing data
  - List Deletion – deleting data row
  - Mean Imputation – replacing with metric mean
  - Regression Imputation – fitting curve, for time based metrics
  - Siminar Response Pattern Imputation – for multidimensional data
  - Multiple Imputation – filling with values of random other classes



De – correlation of metrics

- Interrelations between metrics complicate drawing conclusions
  - Metrics are high correlated
  - Linear factors of high correlatey metrics are variable – constant sum
  - A large class cannot be separated without increasing inheritance / coupling

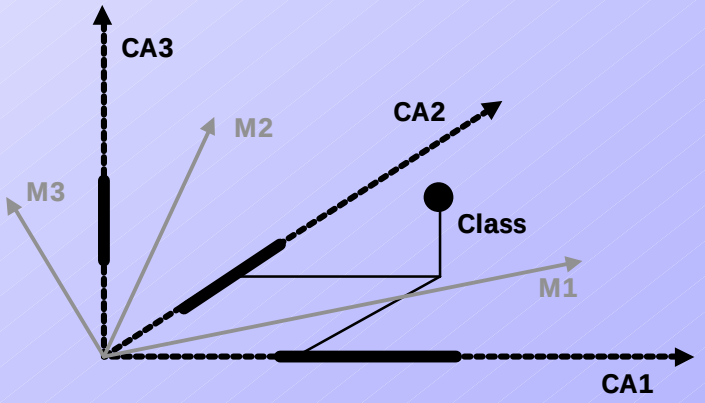
- Principal Components Analysis removes Interrelations
  - Basis technique for generating independent variables
  - Every Principal Component is a linear combination of metrics
  - Factors of linear combination make Principal Components interpretable
  - Eigenvalues of Principal Components describing importance (data variance)

Principal Components Analysis

- Multivariate technique for independent metrics
  - Each Principal Component is a linear combination of metrics

$$y_1 = \sum_{i=1}^{n} a_{1i} * x_i ; \qquad \sum_{i=1}^{n} a_{1i}^2 = 1$$

  - Principal Components describing decreasing data variance
  - Rotation and orthogonalisation in n-dim. Space
  - Axis points to maximum variance



- Interpretable transformation to independent interpretable ‘virtual metrics’

From Principal components to complexity aspects

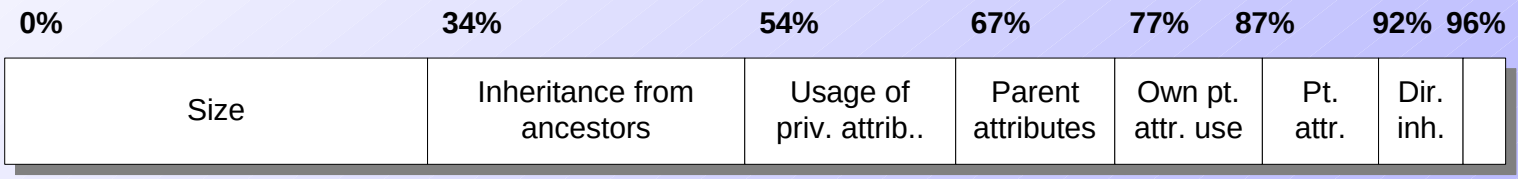
- Results of PCA technique describe aspects and importance
  - Metric factors for linear combination indicate meaning of the Principal Component
  - Eigenvalue indicates importance in the data set

PC / Metric	LOC	DIT	NOM	CBO
1: Factor loading	1.1	0.2	0.9	0.4
1: Contribution	40%	5%	30%	15%
2: Factor loading	-0.1	0.1	0.3	0.5
2: Constribution	5%	5%	30%	40%

- Example for interpretation
  - PC1 size related
  - PC2 coupling related independent to size

Praxis example: technical software

- Component of railway operation system surveyed
  - Language C++
  - Approx. 800 classes, 15 metrics
  - Outlier and missing values inspected
  - Only components with Eigenvalue >0.5 selected



- Toplists for all classes and all complexity aspects
  - Basis for selection of reengineered classes

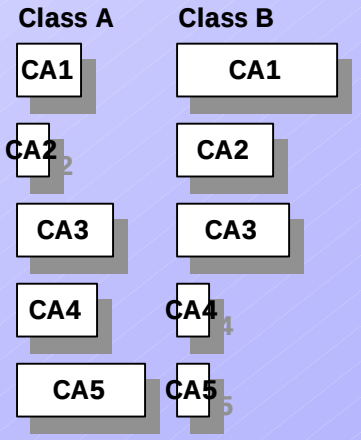
Usage and prospects

Characteristics

- Overview on complexity aspects of object-oriented Software
- Interpretation through descriptive vectors and metrics
- Improved system view with independent aspects
- Usable for object oriented source code and (shortened) UML-Design

Further usage

- Independent ‘virtual metrics’ for empirical modeling techniques
- Reduced calculation number



Complexity comparision between system classes for test, integration order or reengineering

Discussion