Some Patterns and Processes for Refactoring

Rushikesh K Joshi & Students

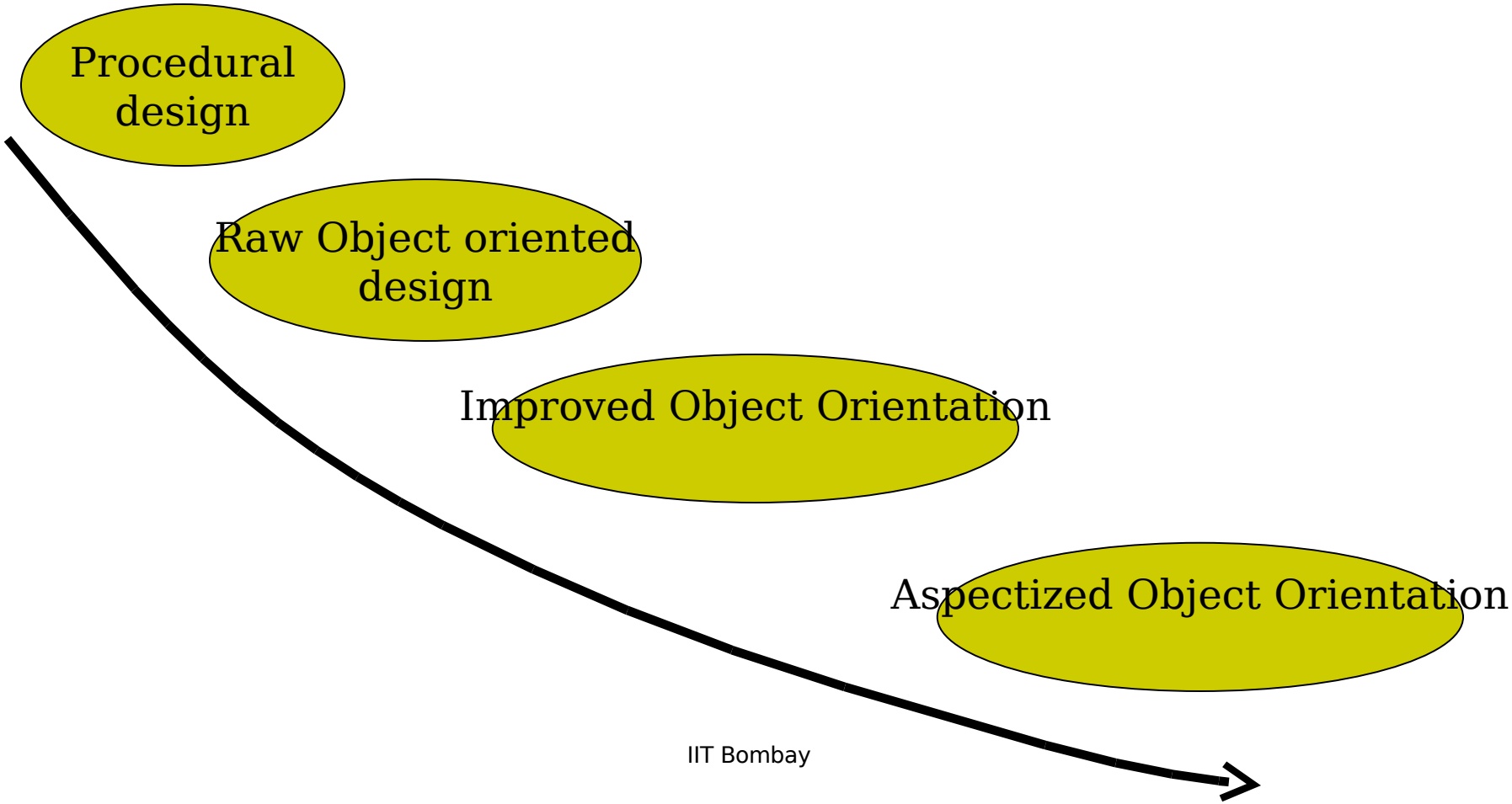
Department of Computer Science and Engineering

IIT Bombay

Refactoring for Design Improvement

- Our focus has been on design improvement
  - As opposed to reengineering for functional improvement
- Masters/PhD work in following 3 dimensions:
  - Improving object oriented designs
  - Aspectization of object orientation
  - Objectification of procedural code

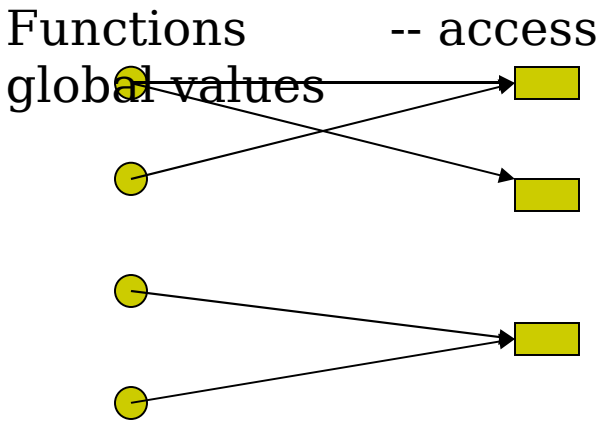
Bird’s Eyeview



Procedural → Object Orientation

- Subject Dimensions
  - Control abstractions:
    - Procedures
    - Pure functions
  - Data Abstractions
    - Primitive types
    - Simple structures
    - Complex Structures
  - Placements/scoping of values and accessibility
    - Global Vs. local
    - Constants, variables
  - Interactions between data and control abstractions
    - Parameter passing
    - Global accesses
    - Static values across function invocations
    - Read/Write accesses

Global Value based Object Identification techniques

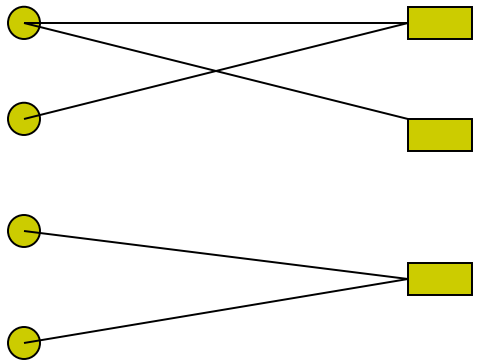


Look for connected components
Each component forms a class

Some Problems:
1. common method such as initializer of globals turns the graph into one big class
2. Global constants may produce noisy collection

Type Based Identification

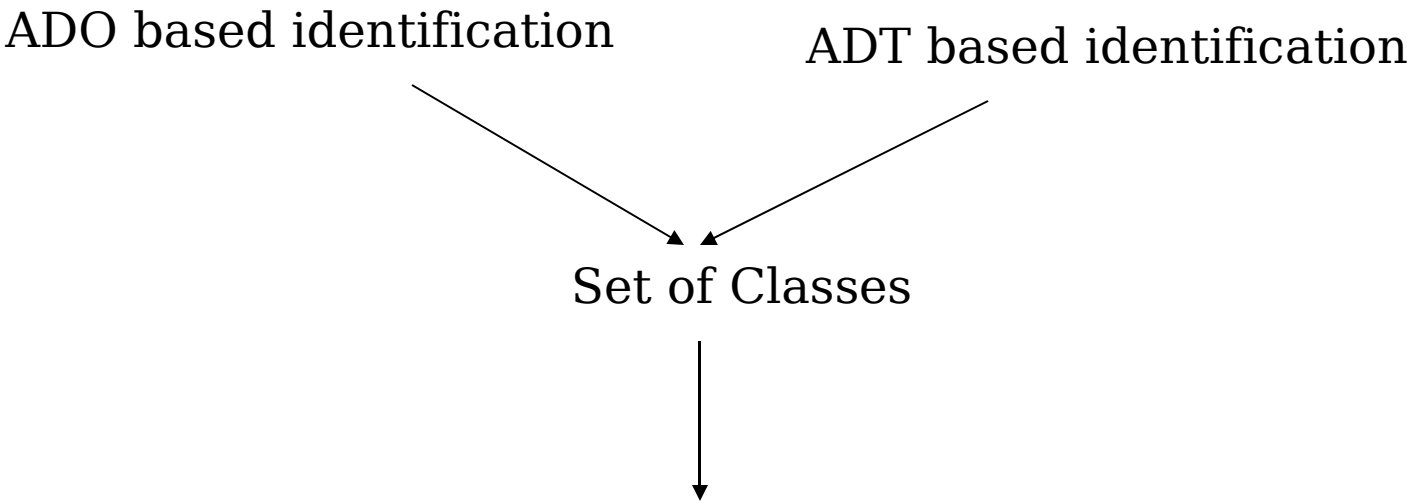
Functions --- arg --- types



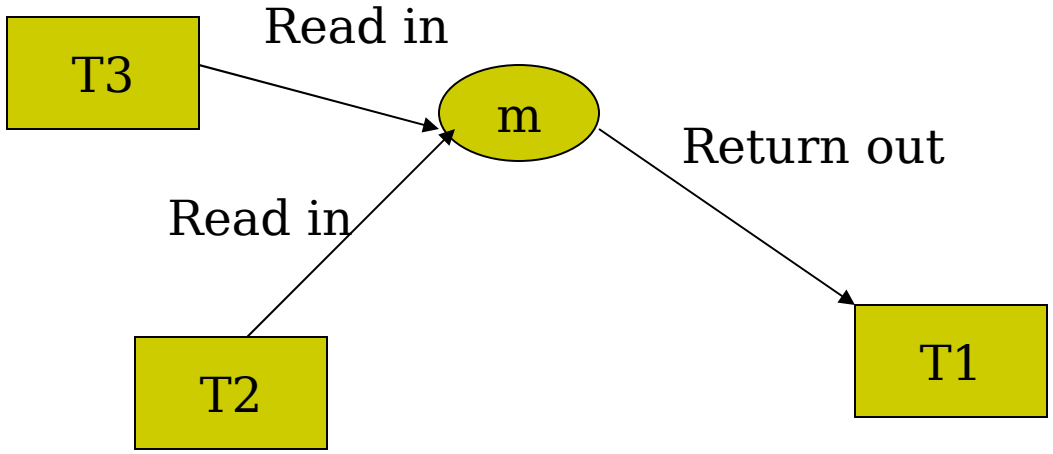
Connected components are classes

Problems:
1.Primitive types are common to many methods – results in noisy classificati
2. No distinction between access modes is made

LCOM based repartitioning on classes [MTech thesis of Ashish Vanarase 2005]



Constructor Identification Technique[in Nishit Desai's M.Tech thesis, 2006]



Open Problems

- Heuristics/metrics for LCOM-based partitioning
- Inheritance Structuring
- Part-Whole Analysis

Project Ox

- Linux kernel 2.6.14 as Testbed
- Experiments on IPC
  - Msg.c
  - Sem.c
  - Shm.c
- A bootable version of kernel in which msg.c is objectified

Objects → Better Objects

- Subject Dimensions:
  - Classes
    - Member functions
    - Attributes
  - Member Functions
    - parameter types
    - Local attributes
    - Local accesses
    - External accesses
  - Dynamics
    - Switch statements
    - Decision boxes

Metric Based Refactoring Techniques[Padmaja, research topic]

- Object Oriented Metrics
  - Mostly at class level i.e. per class
- Refactoring patterns often need attribute level or method level analysis
- Distinction between Macroscopic and Microscopic Metrics
- Need for new metrics from refactoring point of view
- Data model for Structural Representation
  - From the point of view of microscopic and macroscopic metrics

Experiments: Spellchecker and *ASC academic system* code

- Spellchecker Application code refactored with the help of newly defined metrics [Padmaja 2006]
  - Coupling between object reduced
  - LOC remained same,
- An online system in the institute (3200 LOC) [Naval's MTech thesis 2005]
  - Redundant code was a major problem
  - Manual analysis was done for applying fowler's patterns
  - About 36% code was removed

Current Projects

- MJ
  - A tool for computing Metrics for Java Applications
- Structural Representation Model (SRM)
  - Data Model (XML based)
  - language independent
  - Java to SRM tool
  - Metric suites on SRM

Good Objects → Aspects

- Before method call
- After method calls
- parameters passed down the chain
- Exceptions
- contract enforcements
- default implementations of interfaces
- Features/Concerns

Good Objects → Aspects

- Post Aspect Refactoring
  - Move field to aspect
  - Move method to aspect
  - Split constructor etc.
- ASC Application: Code further reduced by 3% (aspect code adds up)
- JSh (90+ classes)
  - Analysis for 15 classes: exception handling classes, default implementations: about 24 aspects, from original class method count came down from 81 to 19– but LOC was just about the same! (1139-1134)

Pointers for New Work..

- Manual analysis of code, writing habits of programmers
- Aspectization patterns
- Aspect Mining techniques
- Possible use of aspects in forward engineering