Welcome to SENG 371 Software Evolution Spring 2013

A Core Course of the BSEng Program

Hausi A. Müller, PhD PEng
Professor, Department of Computer Science
Associate Dean Research, Faculty of Engineering
University of Victoria

Announcements

- Labs
  - One more lab this week
- Final exam
  - Sat, April 13 — 7:00 -10:00 pm
- Last lecture
  - Thu, April 4
  - Review and wrap-up
- Today
  - AntiPattern plays
  - Teaching evaluations
- Marking
  - A2 marks are posted
- Assignment 3
  - Due Thu, April 4
  - Part I — Define software evolution terms
  - Part II — Investigate two AntiPatterns — Vendor-Lock-In — Analysis Paralysis
  - Part III — Refactoring in IBM Eclipse and MS Visual Studio and Blob AntiPattern
  - Cite your sources
  - Submit by e-mail to seng371@uvic.ca

2

Reading Assignment

- Murphy, Notkin, Lan: An empirical study of static call graph extractors, *ACM Transactions on Software Engineering and Methodology (TOSEM)* 7(2):158-191 (1998)
  - http://dl.acm.org/citation.cfm?id=279314
- Müller, Jahnke, Smith, Storey, Tilley, Wong: Reverse Engineering: A Roadmap, in *The Future of Software Engineering*, pp. 47-60 (2000)
  - http://dl.acm.org/citation.cfm?id=336526
- Storey: Theories, tools and research methods in program comprehension: past, present and future, *Software Quality Journal* 14:187-208 (2006)
  - http://webhome.cs.uvic.ca/~chisel/pubs/storey-pc-journal.pdf
- Brown, Malveau, McCormick III, Mowbray: *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis,* John Wiley (1998)
- AntiPatterns Tutorial and Website
  - http://www.antipatterns.com/briefing/index.htm
  - http://www.antipatterns.com

3

Software AntiPatterns

http://en.wikipedia.org/wiki/The_Comedy_of_Errors

Overview

- Motivation
- Reference model
- Software Development AntiPatterns
- Software Architecture AntiPatterns
- Software Management AntiPatterns
- Summary

5

AntiPatterns and Software Evolution

- How do you compare/evaluate software development job offers
- Premise
  - Recognition of AntiPatterns will make you a better software engineer
  - Refactoring AntiPatterns present in a system and/or project will result in a better, more successful, less risky software reengineering project

6

AntiPattern Categories

- Development AntiPatterns
  - LavaFlow, BoatAnchor, GoldenHammer, Poltergeists, SpaghettiCode, Blob, VendorLockIn, WalkingThroughaMineField
- Architectural AntiPatterns
  - SwissArmyKnife, DesignByCommittee, StovePipe, ReinventTheWheel
- Management AntiPatterns
  - AnalysisParalysis, Corncob, DeathByPlanning, MushroomManagement
- AntiPatterns apply to software construction as well as software evolution
- Anti Patterns catalog
  - http://c2.com/cgi/wiki?AntiPatternsCatalog

7

Group Assignment
An AntiPattern "Comedy of Errors" (Play)

- Groups of 4 students
- Pick an AntiPattern
- Develop a play to enact the AntiPattern

- Perform the play in class next week
  - Make sure all group members are involved—ideally equally
  - Include props if need be
  - Practice the play (!)
  - 5 mins for play

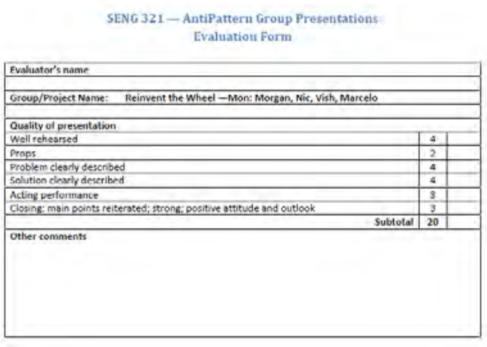
http://en.wikipedia.org/wiki/The_Comedy_of_Errors

8

Pick your play to be performed

- Reinvent the Wheel
  - Mon: Morgan, Nic, Vish, Marcelo
- Design By Committee
  - Mon: Michael, Y, Sam, Mackenzie
- Mushroom Management
  - Mon: Daniel, Brad, Dave, George
- Boat Anchor
- Stovepipe
- Architecture By Implication
- Warm Bodies
- Swiss Army Knife
- Spaghetti Code
- Blob
- Wolf Ticket
- Corncob
  - Thu: Geoff, Adam, Scott, Justin
- Golden Hammer
  - Thu: Rob, Ian, Kai, Saleh
- Walking through a Minefield
  - Thu: Jordan, Amanda, Brandon, Romil
- Poltergeists
  - Thu: Curtis, Mikko, Paul, Allan
- The Grand Old Duke of York
- Dead End
- Cut-and-Paste Programming
- Death by Planning

9



SENG 321 — AntiPattern Group Presentations
Evaluation Form

Evaluator's name		
Group/Project Name: Reinvent the Wheel —Mon: Morgan, Nic, Vish, Marcelo		
Quality of presentation		
Well rehearsed	4	
Props	2	
Problem clearly described	4	
Solution clearly described	4	
Acting performance	3	
Closing: main points reiterated; strong; positive attitude and outlook	3	
Subtotal	20	
Other comments		

10

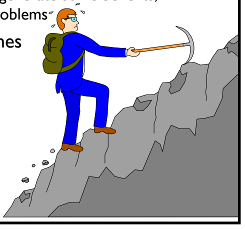
AntiPatterns

- A method for efficiently mapping a general situation to a specific class of solutions
- Provide real-world experience in recognizing recurring problems in the software industry and provide a detailed remedy for the most common predicaments
- Provide a common vocabulary for identifying problems and discussing solutions

11

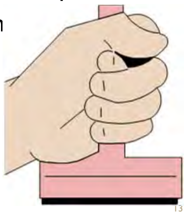
Design Pattern

- Problem
  - Context
  - Applicable design forces
- The role of the solution
  - To resolve the design forces to generate some benefits, consequences, and follow-on problems
- Must occur at least three times



Template

- A consistent outline for the pattern documentation that ensures consistent and adequate coverage of the solution, design forces, and other consequences
- Justification of the pattern and prediction of its consequences



13

Essence of an AntiPattern

- Two solutions instead of a problem and a solution
  - Problematic solution which generates negative consequences
  - Refactored solution, a method to resolve and reengineer the AntiPattern
- A pattern in an inappropriate context

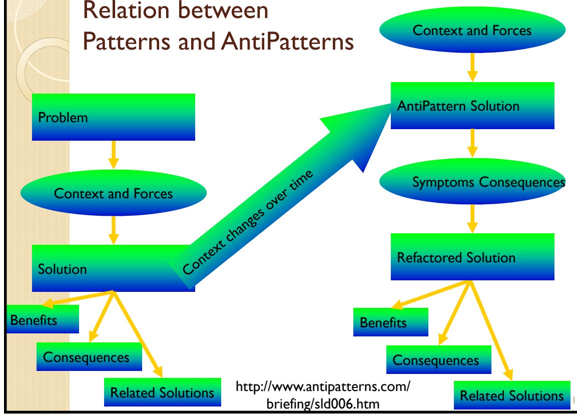
14

Relation between Patterns and AntiPatterns

- Design patterns often evolve into an AntiPattern
- Procedural programming was a great design pattern in the 60's and 70's
- Today it is an AntiPattern
- Object-oriented programming is today a practiced pattern …

15

Relation between Patterns and AntiPatterns



http://www.antipatterns.com/briefing/sld006.htm

16

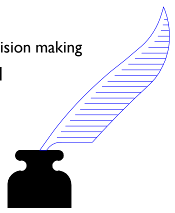
Refactoring: A Useful AntiPattern

- An approach for evolving the solution into a better one
- This process of change, migration, or evolution is called refactoring in the AntiPattern community

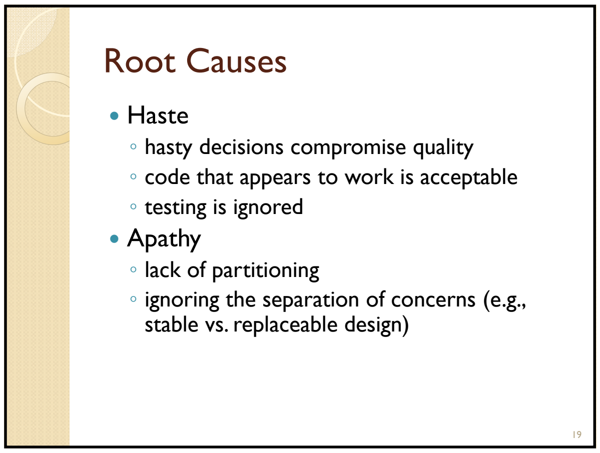
17

Reference Model

- Root causes
  - provide fundamental context for the AntiPattern
- Primal forces
  - are the key motivators for decision making
- Software design-level model
  - define architectural scales; each pattern has a most applicable scale



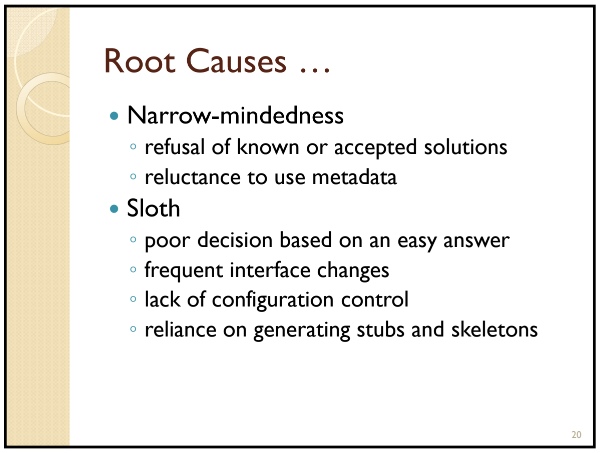
18



Root Causes

- Haste
  - hasty decisions compromise quality
  - code that appears to work is acceptable
  - testing is ignored
- Apathy
  - lack of partitioning
  - ignoring the separation of concerns (e.g., stable vs. replaceable design)

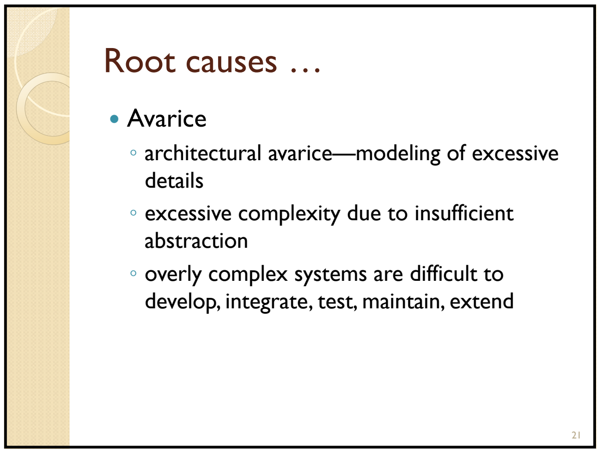
19



Root Causes …

- Narrow-mindedness
  - refusal of known or accepted solutions
  - reluctance to use metadata
- Sloth
  - poor decision based on an easy answer
  - frequent interface changes
  - lack of configuration control
  - reliance on generating stubs and skeletons

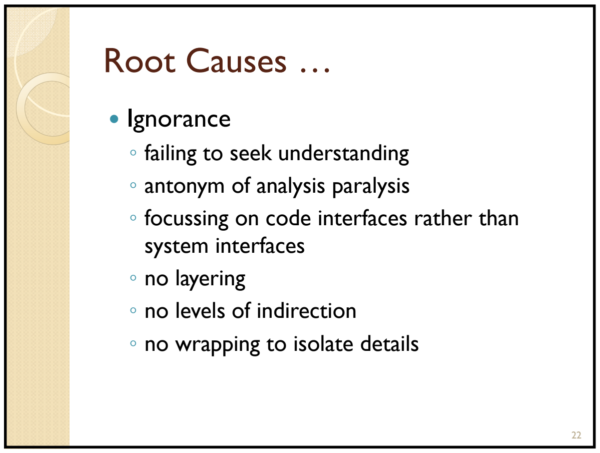
20



Root causes …

- Avarice
  - architectural avarice—modeling of excessive details
  - excessive complexity due to insufficient abstraction
  - overly complex systems are difficult to develop, integrate, test, maintain, extend

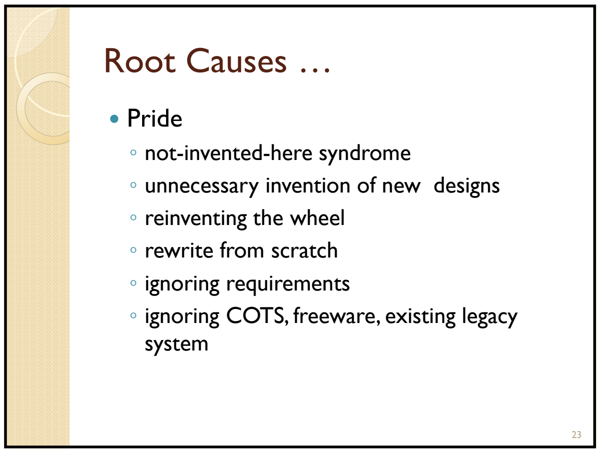
21



Root Causes …

- Ignorance
  - failing to seek understanding
  - antonym of analysis paralysis
  - focussing on code interfaces rather than system interfaces
  - no layering
  - no levels of indirection
  - no wrapping to isolate details

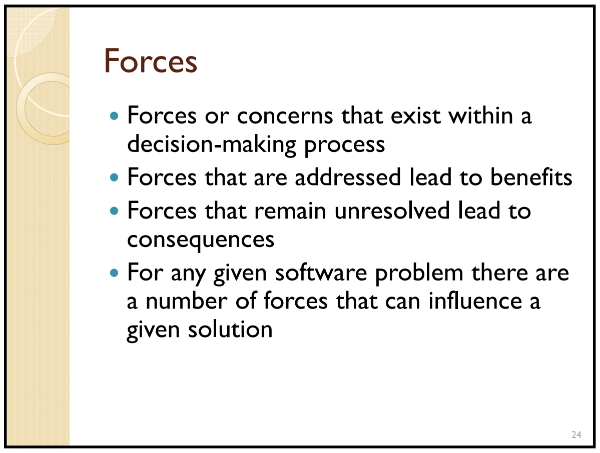
22



Root Causes …

- Pride
  - not-invented-here syndrome
  - unnecessary invention of new designs
  - reinventing the wheel
  - rewrite from scratch
  - ignoring requirements
  - ignoring COTS, freeware, existing legacy system

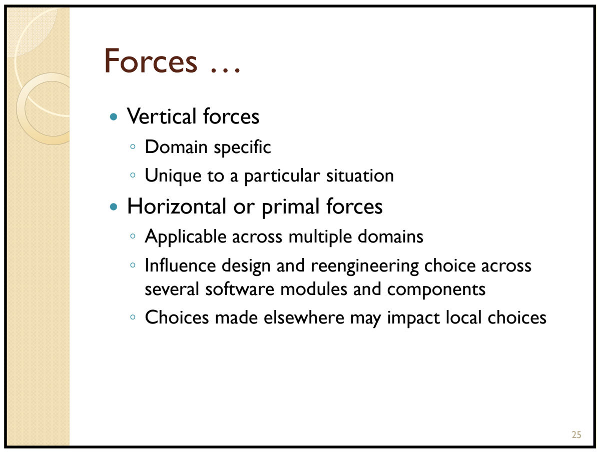
23



Forces

- Forces or concerns that exist within a decision-making process
- Forces that are addressed lead to benefits
- Forces that remain unresolved lead to consequences
- For any given software problem there are a number of forces that can influence a given solution

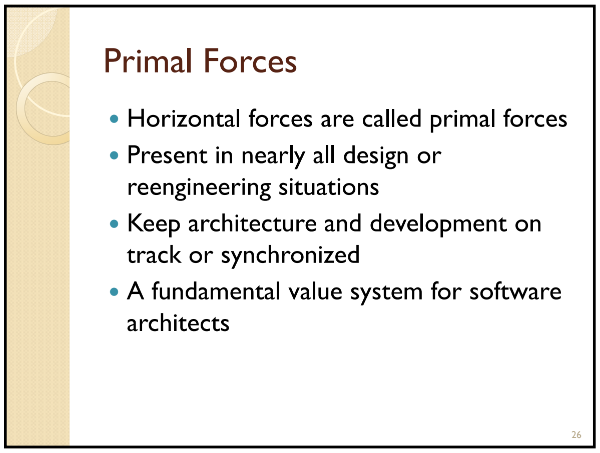
24



Forces …

- Vertical forces
  - Domain specific
  - Unique to a particular situation
- Horizontal or primal forces
  - Applicable across multiple domains
  - Influence design and reengineering choice across several software modules and components
  - Choices made elsewhere may impact local choices

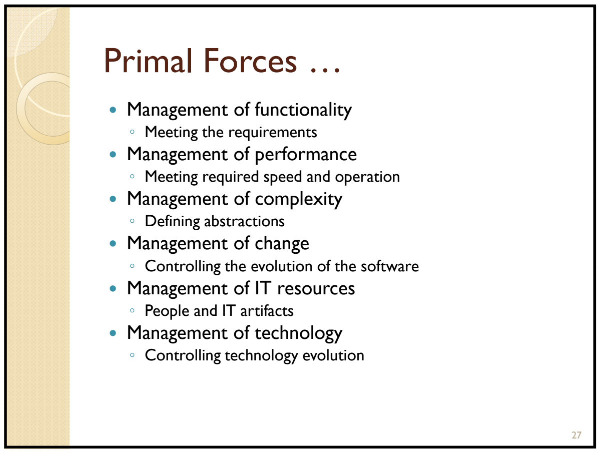
25



Primal Forces

- Horizontal forces are called primal forces
- Present in nearly all design or reengineering situations
- Keep architecture and development on track or synchronized
- A fundamental value system for software architects

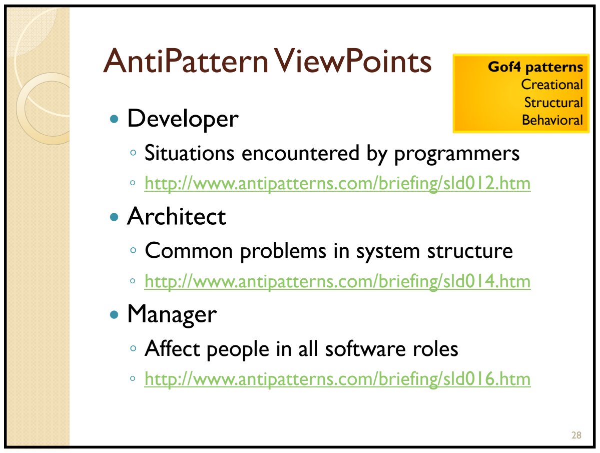
26



Primal Forces …

- Management of functionality
  - Meeting the requirements
- Management of performance
  - Meeting required speed and operation
- Management of complexity
  - Defining abstractions
- Management of change
  - Controlling the evolution of the software
- Management of IT resources
  - People and IT artifacts
- Management of technology
  - Controlling technology evolution

27

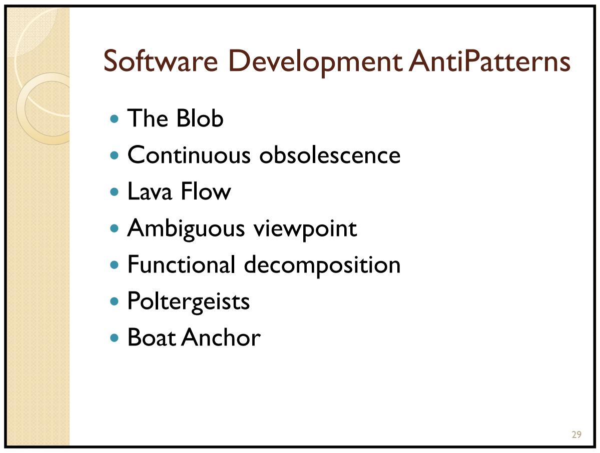


AntiPattern ViewPoints

Gof4 patterns
Creational
Structural
Behavioral

- Developer
  - Situations encountered by programmers
  - http://www.antipatterns.com/briefing/sld012.htm
- Architect
  - Common problems in system structure
  - http://www.antipatterns.com/briefing/sld014.htm
- Manager
  - Affect people in all software roles
  - http://www.antipatterns.com/briefing/sld016.htm

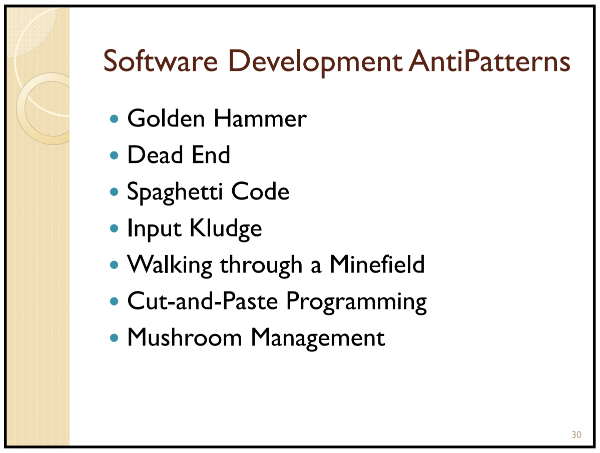
28



Software Development AntiPatterns

- The Blob
- Continuous obsolescence
- Lava Flow
- Ambiguous viewpoint
- Functional decomposition
- Poltergeists
- Boat Anchor

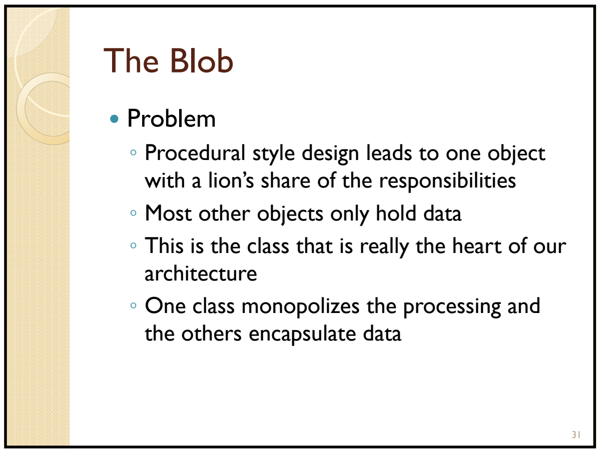
29



Software Development AntiPatterns

- Golden Hammer
- Dead End
- Spaghetti Code
- Input Kludge
- Walking through a Minefield
- Cut-and-Paste Programming
- Mushroom Management

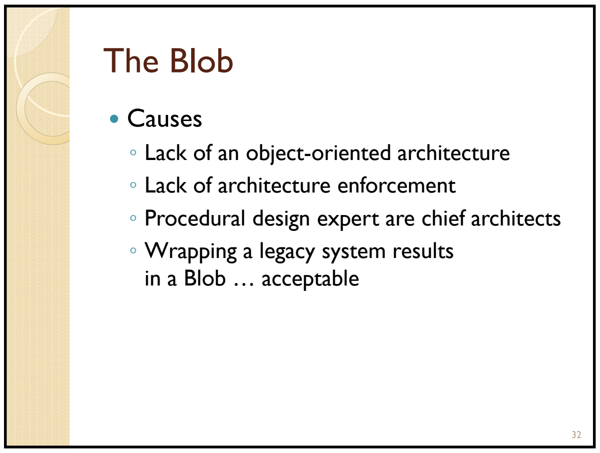
30



The Blob

- Problem
  - Procedural style design leads to one object with a lion's share of the responsibilities
  - Most other objects only hold data
  - This is the class that is really the heart of our architecture
  - One class monopolizes the processing and the others encapsulate data

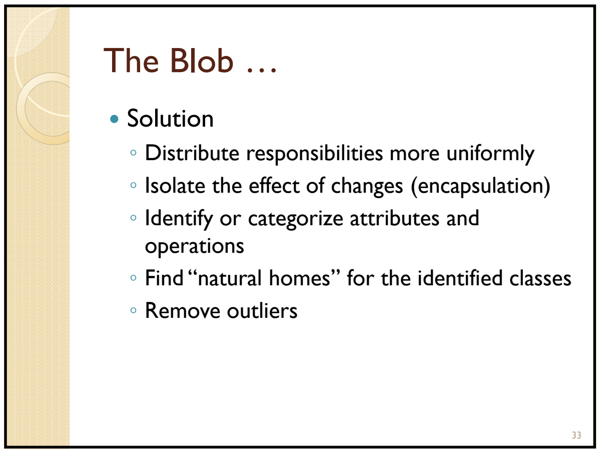
31



The Blob

- Causes
  - Lack of an object-oriented architecture
  - Lack of architecture enforcement
  - Procedural design expert are chief architects
  - Wrapping a legacy system results in a Blob … acceptable

32



The Blob …

- Solution
  - Distribute responsibilities more uniformly
  - Isolate the effect of changes (encapsulation)
  - Identify or categorize attributes and operations
  - Find "natural homes" for the identified classes
  - Remove outliers

33