AntiPatterns

Lecturer: Adel Vahdati

- An AntiPattern describes a commonly occurring solution to a problem that generates decidedly negative consequences
- The AntiPattern may be the result of a manager or developer
  - not knowing any better,
  - not having sufficient knowledge or experience in solving a particular type of problem, or
  - having applied a perfectly good pattern in the wrong context.

- **AntiPatterns** are presented from three perspectives – developer, architect, and manager:
  - **Development AntiPatterns:** comprise technical problems and solutions that are encountered by programmers.
  - **Architectural AntiPatterns:** identify and resolve common problems in how systems are structured.
  - **Managerial AntiPatterns:** address common problems in software processes and development organizations.

AntiPatterns: Development

- **The Blob:** Procedural−style design leads to one object with most of the responsibilities, while most other objects only hold data or simple operations.
- **Lava Flow:** Dead code and forgotten design information is frozen in an ever-changing design.
- **Functional Decomposition:** The output of nonobject−oriented developers who design and implement an application in an object−oriented language.
- **Ambiguous Viewpoint:** Object-oriented analysis and design (OOA&D) models that are presented without clarifying the viewpoint represented by the model.

AntiPatterns: Development

- **Golden Hammer:** A familiar technology or concept applied obsessively to many software problems.
- **Spaghetti Code:** Ad hoc software structure makes it difficult to extend and optimize code.
- **Cut–and–Paste Programming:** Code reused by copying source statements leads to significant maintenance problems.
- **Mushroom Management:** Keeping system developers isolated from the system's end users.