AntiPatterns

Lecturer: Adel Vahdati

AntiPatterns

- 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: Viewpoints

- **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.

AntiPatterns: Architectural

- **Stovepipe System/Enterprise:** Subsystems/systems are integrated in an ad hoc manner using multiple integration strategies and mechanisms.
  - A "stovepipe system" (also called a stovepipe architecture) is a term in software engineering and systems design that refers to a system that is self-contained, isolated, and lacks integration with other systems.
  - It is usually built independently for a specific purpose or department and does not share data, functionality, or infrastructure with other systems.
- **Cover Your Assets:** Document-driven software processes that produce less-than-useful requirements and specifications because the authors evade making important decisions.

AntiPatterns: Architectural

- **Vendor Lock-In:** Vendor Lock-In occurs in systems that are highly dependent upon proprietary architectures.
- **Architecture by Implication:** the lack of architecture specifications for a system under development.
  - It refers to a situation where the architecture of a system is not explicitly designed or documented, but instead emerges implicitly from the accumulation of decisions made during development.
- **Design by Committee:** Design by Committee creates overly complex architectures that lack coherence.
- **Swiss Army Knife:** An excessively complex interface. The designer attempts to provide for all possible uses of the class. In the attempt, he or she adds a large number of interface signatures in an attempt to meet all possible needs.

AntiPatterns: Architectural

- **Reinvent the Wheel:** The pervasive lack of experience transfer between software projects leads to substantial reinvention.
- **The Grand Old Duke of York:** describes a cycle of development where teams or leaders initiate effort (e.g., start architectural changes, redesign, or refactoring), only to reverse course later—resulting in no net progress, wasted effort, and frustrated teams.

"Oh, the grand old Duke of York,
He had ten thousand men,
He marched them up to the top of the hill,
And he marched them down again."

AntiPatterns: Management

- **Analysis Paralysis:** Striving for perfection and completeness in the analysis phase leading to project gridlock and excessive work on requirements/models.
- **Viewgraph Engineering:** On some projects, developers become stuck preparing viewgraphs and documents instead of developing software.
- **Death by Planning:** Excessive planning for software projects leading to complex schedules that cause downstream problems.
- **Fear of Success:** Often occurs when people and projects are on the brink of success. Some people begin to worry obsessively about the kinds of things that can go wrong.

AntiPatterns: Management

- **Corncob:** Difficult people frequently obstruct and divert the software development process.
- **Intellectual Violence:** Intellectual violence occurs when someone who understands a theory, technology, or buzzword uses this knowledge to intimidate others in a meeting situation.
- **Project Mismanagement:** Inattention to the management of software
- development processes causing directionless-ness and other symptoms.

References

- Brown, W. J., Malveau, R. C., McCormick, H., Mowbray, T., Antipatterns: Refactoring Software, Architectures, and Projects in Crisis. Wiley, 1998.
- Ramsin, Raman. "Home." Department of Computer Science and Engineering, Sharif University of Technology. Accessed February 15, 2025. https://sharif.edu/~ramsin/index.htm.