**POM 18 – Project Management Antipatterns**

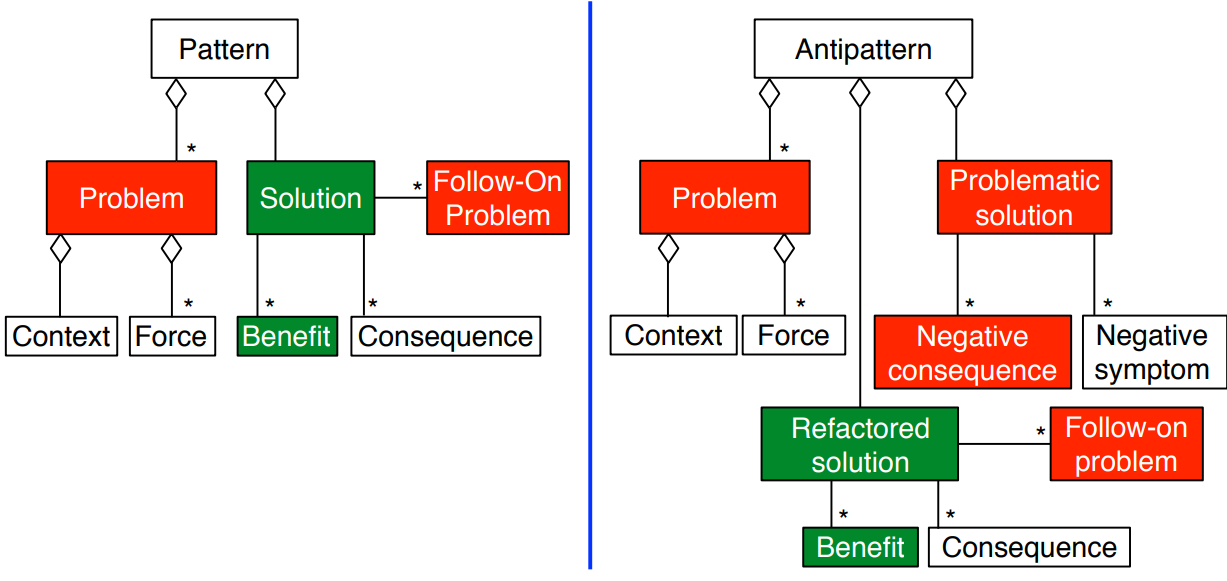
*Pattern vs Antipatterns:*

**Pattern definition**

* A pattern has two parts: problem and solution
  + Problem class: **context** set of **forces**
  + Solution: resolves forces with **benefits** and **consequences**
  + solution must be applicable to **more than one** specific problem
* Solutions usually generate **follow-on problems**
  + Follow-on problems may lead to the applicability of other patterns
* **Patterns can evolve into antipatterns when change occurs**
  + Change of requirements, change of project parameters, change of methodology

**Antipattern definition**

* Antipatterns identifizieren und kategorisieren die **häufigsten Fehler in der Software-Praxis**
* "Ein Antipattern ist etwas, das **wie eine gute Idee aussieht**, aber bei Anwendung **schlecht zurückschlägt**"
* Synonyms (bad practice, bad idea, typical design error, …)
* An antipattern consists of 1 problem and 2 solutions
  + Problematic solution: generates overwhelming **negative consequences**
  + Refactored solution: describes how problematic solution can be **reengineered** to avoid these negative consequences and **lead to benefits**, can also lead again to **follow-on problems**



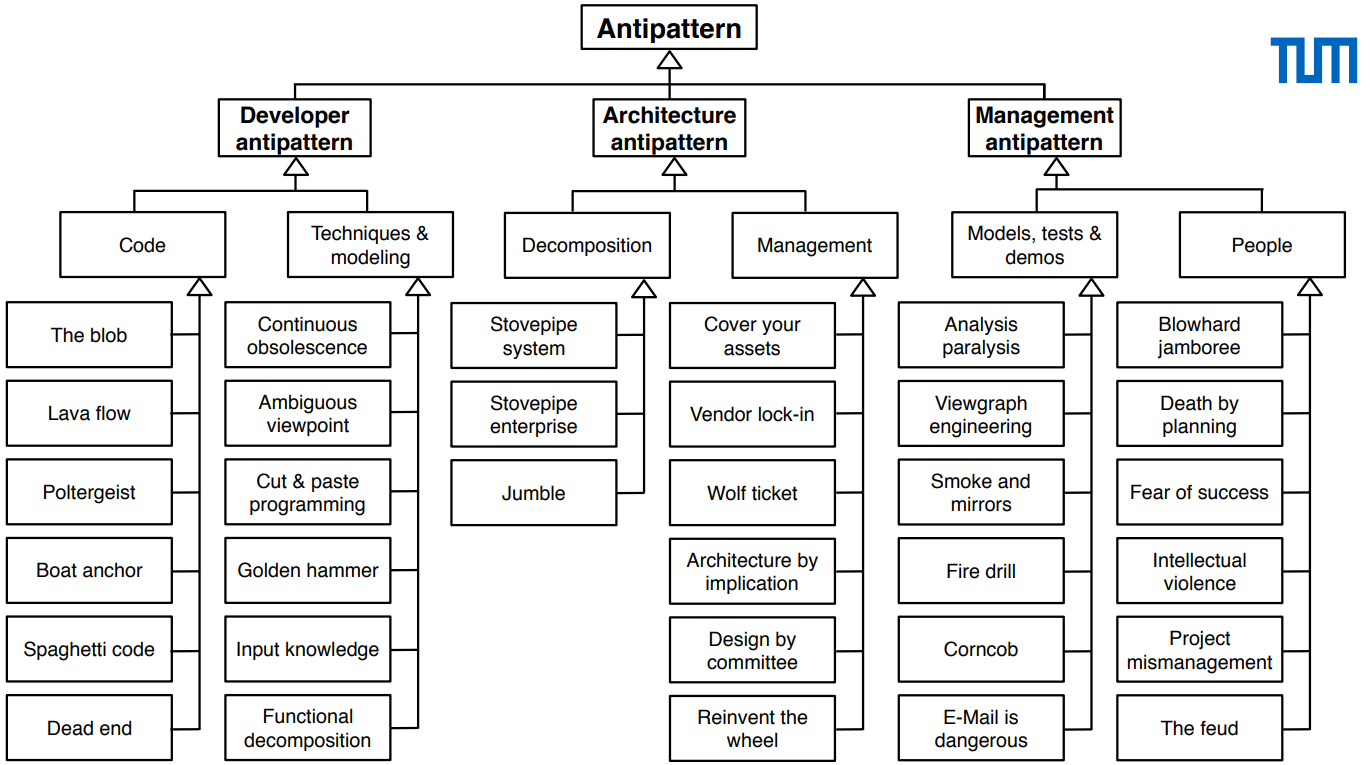
**The seven deadly sins in software practice**

1. Haste [Eile] (hasty decisions)
2. Pride [Stolz] (not willing to adopt anything from the outside)
3. Apathy [Gleichgültigkeit] (not caring about a problem, unwillingness to attempt a solution)
4. Sloth [Faulheit] (poor decisions, “easy” answers)
5. Ignorance (failure to seek understanding)
6. Avarice [Geiz] (no use of abstractions and modeling of details)
7. Narrow-mindedness [Engstirnigkeit] (“why reuse? I only have to solve one problem”)

**Antipattern taxonomy**

3 types of antipatterns

* **Development antipatterns** (functional decomposition)
  + Focus on the viewpoint of the software developer
  + **Issues**: software refactoring, modification of source code
* **Architecture antipatterns** ()
  + Focus on the viewpoint of the software architect
  + **Issues**: partitioning of subsystems and components, platform independent definition of interfaces
* **Management antipatterns** (analysis paralysis, corncob, death by planning)
  + Focus on the viewpoint of the software project manager
  + **Issues**: software project organization, software project management, human communication



**Analysis paralysis**

* Spending excessive time in requirements elicitation and analysis
* Example: plan driven software development (main goal minimize risk through careful upfront planning, no iterations, just one pass)

**Functional decomposition**

* Design and implementation style of developers, who got their initial training in structured analysis methods and imperative languages, with no or little experience in OO methods and OO languages

**Corncob**

* Difficult people who usually create additional stress
* 7 types of difficult people:
  1. Hostile Aggressives
  2. Indecisives [unentschlossene]
  3. Whiners
  4. Negativists
  5. Clams [schweigsame]
  6. Bulldozers
  7. Superagreeables

**Death by planning**

* Too much planning, lack of an up-to-date project plan

**Mushroom**

* Bad understanding of requirements, management isolates developers from the system’s end users