

Die 7 Todsünden der Softwareentwicklung

Die christlichen Todsünden

- Die christlichen 7 Todsünden gehen auf Evagrius Ponticus zurück
- Mönch im 4. Jahrhundert n. Chr.
- Ursprung 8 Dämonen: **Völlerei, Unkeuschheit, Habsucht, Zorn**, Trübsinn, Faulheit, Ruhmsucht, Stolz
- Ursprüngliche 8 Dämonen, von Papst Gregor im 6. Jahrhundert auf die mythische Zahl 7 reduziert
 - aus Trübsinn und Faulheit wurde **Trägheit**
 - aus Ruhmsucht und Stolz wurde **Hochmut**
 - **Neid** kam hinzu

Aufbau der Todsündenfolien

- Name der Todsünde
- Beschreibung in Stichworten mit Aussagebeispielen
- Resultierende Antipattern

Was sind Antipattern

- Antipattern sind Negativbeispiel in der Softwareentwicklung mit Lösungsanleitungen
- Das Gegenteil sind Pattern, also Lösungsansätze für Probleme
- Antipattern müssen mindestens 3 mal beobachtet worden sein
- Antipattern sind lehrreicher als Pattern
- Antipattern und Lösungen werden nicht besprochen, nur erwähnt

1. Unangebrachte Hast

- Hastige Entscheidungen führen zu Fehlern
- Unrealistische Zeitvorgaben
- Qualität wird während der Projektlaufzeit immer schlechter
 - Auf Tests wird verzichtet
- Fachwissen fehlt
- Entwurfskomplexität wird unterschätzt

Antipattern

- Der Blob (The God Class)
- Stovepipesysteme

2. Desinteresse

- Mangelnde Sorgfalt bei der Lösung von Problemen
- Teilnahmslosigkeit verhindert saubere Gliederung
- Schlechtes Schnittstellendesign
- „Wiederverwenden? Wer wird diesen verdammten Code jemals wiederverwenden?“

Antipattern

- Stovepipe Enterprise Systeme (Islands of Automation)
- Vendor Lock-In (Product-Dependent Architecture)

3. Engstirnigkeit

- Verweigerung gegenüber Lösungen die sich als praktisch (Pattern) oder unpraktisch (Antipattern) erwiesen haben
- „Bei uns gibt es nichts zu konfigurieren. Das ist alles im Code“
- „Unsere Datenbank ist unsere Architektur“
- „Vielleicht hätten wir hierfür lieber Excel verwenden sollen“

Antipattern

- Golden Hammer (Head-in-the Sand)
- Stovepipe Enterprise (Islands of Automation)

4. Faulheit

- Schlechte Entscheidungen auf einfache Fragen
- Schlampiges Schnittstellendesign
- Schlechte Dokumentation
- „Diese Klasse ist das Herzstück unserer Applikation“

Antipattern

- Cut-and-Paste Programmierung
- Lava Flow (Dead Code)
- Spaghetti Code

5. Geiz

- Mangelnde Abstraktion führt zu ausufernder Komplexität
Dies führt zu:
 - hohen Entwicklungskosten
 - hohen Testkosten
 - hohen Pflege- und Erweiterungskosten
- R&D Code in Produkt übernehmen

Antipattern

- Lava Flow (Dead Code)
- Stovepipe-System

6. Ignoranz

- Form intellektueller Faulheit
- „Ich bin nicht ganz sicher was diese Klasse macht, aber es ist sicher sehr wichtig.“
- „Unser Problem ist einmalig“
- „Ich habe wirklich nicht die Zeit das alles zu lesen. Schreibe sie mir eine Zusammenfassung. Vielleicht eine Seite.“
 - Ignoranten verhindern Veränderungen

Antipattern

- Poltergeist
- Spaghetti Code
- Reinvent the Wheel (Greenfield System)

7. Stolz

- Mangelhafte Unterstützung von Wiederverwendbarkeit
 - „Das ist nicht von uns“
- Einführung neuer Elemente obwohl System dies schon unterstützt.
 - Wohlgemerkt, das eigene System!
- „Ich muss dabei mitmachen... ***(bin ich wichtig, yeah!)***“

Antipattern

- Golden Hammer (Head-in-the Sand)
- Vendor Lock-In
- Design by Committee (Make everybody happy)

Quelle

- Brown, William J.; Malveau, Raphael C.; McCormick, Hays W. „Skip“ II; Mowbray, Thomas J., AntiPatterns: Entwurfsfehler erkennen und vermeiden, mitp-Verlag, 2004
 - ISBN: 3-8266-1479-8
 - <http://www.mitp.de/>

Fragen