Resign Patterns
Ailments of Unsuitable Project-Disoriented Software
by
Michael Duell

Abstract

Anyone familiar with the book of patterns by the Gang of Four [1] knows that the patterns presented in the book represent elegant solutions that have evolved over time. Unfortunately, extracting these patterns from legacy code is impossible, because nobody knew that they were supposed to be using these patterns when they wrote the legacy code. Hence, this work is a catalog of patterns for the masses. The patterns presented here represent abundant solutions that have endured over time. Enjoy reading the patterns, but please don't use them!

1 Cremational Patterns

Below is a list of five cremational patterns.

1.1 Abject Poverty

The Abject Poverty Pattern is evident in software that is so difficult to test and maintain that doing so results in massive budget overruns.

1.2 Blinder

The Blinder Pattern is an expedient solution to a problem without regard for future changes in requirements. It is unclear as to whether the Blinder is named for the blinders worn by the software designer during the coding phase, or the desire to gouge his eyes out during the maintenance phase.

1.3 Fallacy Method

The Fallacy method is evident in handling corner cases. The logic looks correct, but if anyone actually bothers to test it, or if a corner case occurs, the Fallacy of the logic will become known.

1.4 ProtoTry

The ProtoTry Pattern is a quick and dirty attempt to develop a working model of software. The original intent is to rewrite the ProtoTry, using lessons learned, but schedules never permit. The ProtoTry is also known as legacy code.

1.5 Simpleton

The Simpleton Pattern is an extremely complex pattern used for the most trivial of tasks. The Simpleton is an accurate indicator of the skill level of its creator.

2 Destructural Patterns

Below is a list of seven destructural patterns.

2.1 Adopter

The Adopter Pattern provides a home for orphaned functions. The result is a large family of functions that don't look anything alike, whose only relation to one another is through the Adopter.

2.2 Brig

The Brig Pattern is a container class for bad software. Also known as module.

2.3 Compromise

The Compromise Pattern is used to balance the forces of schedule vs. quality. The result is software of inferior quality that is still
late.

2.4 Detonator

The Detonator is extremely common, but often undetected. A common example is the calculations based on a 2 digit year field. This bomb is out there, and waiting to explode!

2.5 Fromage

The Fromage Pattern is often full of holes. Fromage consists of cheesy little software tricks that make portability impossible. The older this pattern gets, the riper it smells.

2.6 Flypaper

The Flypaper Pattern is written by one designer and maintained by another. The designer maintaining the Flypaper Pattern finds herself stuck, and will likely perish before getting loose.

2.7 ePoxy

The ePoxy Pattern is evident in tightly coupled software modules. As coupling between modules increases, there appears to be an epoxy bond between them.

3 Misbehavioral Patterns

Below is a list of eleven misbehavioral patterns.

3.1 Chain of Possibilities

The Chain of Possibilities Pattern is evident in big, poorly documented modules. Nobody is sure of the full extent of its functionality, but the possibilities seem endless. Also known as Non-Deterministic.

3.2 Commando

The Commando Pattern is used to get in and out quick, and get the job done. This pattern can break any encapsulation to accomplish its mission. It takes no prisoners.

3.3 Intersperser

The Intersperser Pattern scatters pieces of functionality throughout a system, making a function impossible to test, modify, or understand.

3.4 Instigator

The Instigator Pattern is seemingly benign, but wreaks havoc on other parts of the software system.

3.5 Momentum

The Momentum Pattern grows exponentially, increasing size, memory requirements, complexity, and processing time.

3.6 Medicator

The Medicator Pattern is a real time hog that makes the rest of the system appear to be medicated with strong sedatives.

3.7 Absolver

The Absolver Pattern is evident in problem ridden code developed by former employees. So many historical problems have been traced to this software that current employees can absolve their software of blame by claiming that the absolver is responsible for any problem reported. Also known as It's-not-in-my-code.

3.8 Stake

The Stake Pattern is evident in problem ridden software written by designers who have since chosen the management ladder. Although fraught with problems, the manager's stake in this software is too high to allow anyone to rewrite it, as it represents the pinnacle of the manager's technical achievement.

3.9 Eulogy

The Eulogy Pattern is eventually used on all projects employing the other 22 Resign Patterns. Also known as Post Mortem.

3.10 Tempest Method

The Tempest Method is used in the last few days before software delivery. The Tempest Method is characterized by lack of comments, and introduction of several Detonator Patterns.

3.11 Visitor From Hell

The Visitor From Hell Pattern is coincident with the absence of run time bounds checking on arrays. Inevitably, at least one control loop per system will have a Visitor From Hell Pattern that will overwrite critical data.

4 References

Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

Michael Duell is an Engineer at AG Communication Systems, where his Resign Patterns have been rejected in favor of the Gang of Four Design Patterns.

"Resign Patterns: Ailments of Unsuitable Project-Disoriented Software," The Software Practitioner, Vol. 7, No. 3, May-June 1997, p. 14.