Anti-patterns

- Spaghetti code
- Lasagna code
- Interface bloat
- Call super
- God object
- Busy-waiting
- Coding by exception
- Hard coding
- Magic numbers
- Cargo cult programming

Spaghetti code

Code that has a complex and tangled control structure, especially one using many GOTO statements, exceptions, threads, or other "unstructured" branching constructs.

It is named such because program flow is conceptually like a bowl of spaghetti, i.e. twisted and tangled. Spaghetti code can be caused by several factors, such as continuous modifications by several people with different programming styles over a long life cycle.

Lasagna code

Lasagna code is any program structure characterized by several well-defined and separable layers, where each layer of code accesses services in the layers below through well-defined interfaces. The analogy stems from the layered structure of lasagna, where different ingredients (for example, meat, sauce, vegetables, or cheese) are each separated by strips of pasta.

One common instance of lasagna code occurs at the interface between different subsystems, such as between web application code, business logic, and a relational database. Another common programming technique, alternate hard and soft layers (use of different programming languages at different levels of the program architecture), tends to produce lasagna code. In general, client–server applications are frequently lasagna code, with well-defined interfaces between client and server.

Interface bloat

Also called *fat interfaces* is when a computer interface incorporates too many operations on some data into an interface, only to find that most of the objects cannot perform the given operations.

Call super

Call super is a design pattern in which a particular class stipulates that in a derived subclass, the user is required to override a method and call back the overridden function itself at a particular point.

The overridden method may be intentionally incomplete, and reliant on the overriding method to augment its functionality in a prescribed manner.

God object

Object that knows too much or does too much.

A common programming technique is to separate a large problem into several smaller problems (a divide and conquer strategy) and create solutions for each of them. Once the smaller problems are solved, the big problem as a whole has been solved.

Busy-waiting

Busy-waiting or busy-looping or spinning is a technique in which a process repeatedly checks to see if a condition is true.

Instead of busy-waiting applications must use subscription on events or messages.

Coding by exception

Coding by exception is using these exceptions to handle specific errors that arise to continue the program.

Exceptions can be used to handle the error while the program is running and avoid crashing the system. Exceptions should be generalized and cover numerous errors that arise.

Hardcoding

The software development practice of embedding an input or configuration data directly into the source code of a program or other executable object, or fixed formatting of the data, instead of obtaining that data from external sources or generating data or formatting in the program itself with the given input.

Magic number

The programming practice of using numbers directly in source code.

The use of unnamed magic numbers in code obscures the developers' intent in choosing that number, increases opportunities for subtle errors and makes it more difficult for the program to be adapted and extended in the future.

Replacing all significant magic numbers with named constants makes programs easier to read, understand and maintain.

Cargo cult programming

A style of computer programming characterized by the ritual inclusion of code or program structures that serve no real purpose.

Cargo cult programming can also refer to the results of applying a design pattern or coding style blindly without understanding the reasons behind that design principle.