WHAT IS SPAGHETTI CODE (AND WHY YOU SHOULD AVOID IT)

Spaghetti code is not an endearing term. It's a word that describes a type of code that some say will cause your technology infrastructure to fail. Despite the warnings about spaghetti code, many enterprise businesses are faced with issues that involve the set of processes and errors that make up spaghetti coding. This occurs for a number of reasons but can be detrimental to your overall infrastructure, most experts believe. In this article, we'll talk about spaghetti code as a concept and what it means for your organization.

What is Spaghetti Code?

Spaghetti code is a pejorative piece of information technology jargon that is caused by factors like unclear project scope of work, lack of experience and planning, an inability to conform a project to programming style rules, and a number of other seemingly small errors that build up and cause your code to be less streamlined overtime. Typically, spaghetti code occurs when multiple developers work on a project over months or years, continuing to add and change code and software scope with optimizing existing programming infrastructure.

This usually results in somewhat unplanned, convoluted coding structures that favor GOTO statements over programming constructs, resulting in a program that is not maintainable in the long-run. For businesses, creating a program only to have it become unmaintainable after years of work (and money) has gone into the project costs, IT managers, and other resources. In addition, it makes programmers feel frustrated to spend hours on coding an infrastructure only to have something break, and then they have to sift through years of work, often managed by different developers, to

identify the issue, if they can solve it at all.

For this reason, spaghetti code is considered a nuisance to developers and IT managers, and for enterprise businesses that have to manage their resources, it should be avoided completely.

History of Spaghetti Code

While it's not clear who coined the term "spaghetti code" or when, it was being used to describe a tangled mess of code lacking structure by the late 1970s. In the 80s, the term was used at least once in a whitepaper to describe the model of code and fix that led to the development of waterfall programming. However, the vast majority of books from that era refer to it as a nest of messy code lacking the structure required to scale effectively.

A 1981 coding satire raised the idea that the founders of IBM must have been fond of spaghetti code because of how they developed FORTRAN. In Richard Hamming's book, The Art of Doing Science and Engineering: Learning to Learn, he said:

"If, in fixing up an error, you wanted to insert some omitted instructions then you took the immediately preceding instruction and replaced it by a transfer to some empty space. There you put in the instruction you just wrote over, added the instructions you wanted to insert, and then followed by a transfer back to the main program.

"Thus the program soon became a sequence of jumps of the control to strange places. When, as almost always happens, there were errors in the corrections you then used the same trick again, using some other available space. As a result the control path of the program through storage soon took on the appearance of a can of spaghetti. Why not simply insert them in the run of instructions? Because then you would have to go over the entire program and change all the addresses which referred to any of the moved instructions! Anything but that!"

How Does it End Up in Your Infrastructure?

Spaghetti programming is not clean or structured and flies in the face of scopes and models put together by enterprise businesses; so how does it get there? Spaghetti code occurs when certain conditions are met:

- A programmer or programmers did not take care to finesse the architecture using programming constructs, and instead relied on easier, or less thought out approaches, or just dove into a project and started coding without a plan.
- Development best practices and most streamlined coding languages change over time, so when programming evolves, existing systems should be optimized to keep them clean and structured. When this fails to occur, it creates an ecosystem of spaghetti code.
- While spaghetti code can occur just hours after starting a project, in enterprise business, it often occurs over a longer period where different programmers change hands on a project.
- Spaghetti code can occur when someone lacks experience and they use simpler techniques, like GOTO commands, over more refined ones that secure the structural integrity of program architecture.

In general, spaghetti code is the natural result that happens to businesses when they fail to plan for a reliable architecture over the life of a project.

Spaghetti Code: How to Avoid It

To avoid a tangle of inefficient spaghetti code, programmers must:

- **Be diligent:** First and foremost, diligence and attention to detail are key. A developer must be keenly focused on creating the best architecture for their project and must not rush the architecture.
- **Unit test often:** By performing regular unit tests, you can reduce the likelihood that spaghetti code will occur.
- **Double-check your programmers:** An extra set of eyes can help, if you come across spaghetti code, address it immediately and ask that it be changed.
- **Use lightweight frameworks:** In 2020, there are a number of lean, lightweight frameworks you can implement. By keeping the framework streamlined, you are setting yourself up for more simplified solutions.
- **Implement layers:** In practice, layers help you correct spaghetti code more easily by addressing a single layer instead of an entire program.

Related Code Types

There are a few related types of code you can review for additional learning:

- **Ravioli code:** This is the term for errors in object-oriented code that occur when code is easy to understand in a class but not in the context of the entire project.
- **Lasagna code:** This is a problem that can occur when you use layers to avoid spaghetti code and the layers are so interdependent on one another that a single break in a layer affects the whole project.
- **Pizza code:** If a code architecture is too flat, it's called a pizza code.