**Julio Pochet - Design by Contract**

I chose **Design by Contract** from *The Pragmatic Programmer* because I’ve seen first-hand how unclear expectations in coding can lead to frustrating bugs. Sometimes, a function works fine in one part of a program but completely breaks when used elsewhere. This concept caught my attention because it enforces **clear rules for what a function should receive (preconditions), what it should return (postconditions), and what should always remain the same (invariants)**. These rules make debugging easier, especially in **brownfield projects**, where modifying old code can accidentally create new issues.

The main idea of **Design by Contract** is that functions should have strict agreements about what they do. For example, if a function spreads fire in a forest fire simulation, its contract could state that it **only takes a valid grid where at least one tree is burning** and **only spreads to flammable areas**. Without these rules, the fire could spread randomly, creating unrealistic behavior or even causing the program to crash. Using **Design by Contract** ensures that every part of the system interacts correctly, making the simulation more reliable and maintainable.

A great resource that explains this concept further is *"Design by Contract: Part One"* by Leading Agile. The article explains how Design by Contract helps create more reliable software by enforcing strict input and output expectations. You can check it out here:

[Design by Contract: Part One](https://www.leadingagile.com/2018/05/design-by-contract-part-one/)