**Julio Pochet - The Power of Decoupling in Software**

I chose **Decoupling** from *The Pragmatic Programmer* because I see its usefulness, especially when working with existing code. Right now, I’m modifying a **forest fire simulation**, and if different parts of the code were too tangled up, making changes would be a nightmare. Decoupling helps by keeping things separate—so I can tweak how the fire spreads without messing up the whole program. When working on **brownfield projects**, it's a game-changer where you add new features to old code without breaking everything.

At its core, **decoupling means making different parts of a program less dependent on each other**. Instead of having one big chunk of code that does everything, you break it into smaller, focused pieces. This follows the **Single Responsibility Principle (SRP)**—each program part should only do one thing. For example, in my forest fire simulation, I should be able to update the **fire spread logic** without touching the **visualization**. That way, if I want to make the graphics more detailed, I won’t have to rewrite the entire fire simulation.

Netflix's microservices system is a good example of decoupling in the real world. Instead of running as one massive application, Netflix is split into smaller independent services—one for recommendations, one for streaming, another for user profiles, etc. If they need to update their recommendation algorithm, it won’t break the streaming service. That’s the same idea I’m applying to my project: keeping things modular so future updates are easier.

For more details, I found an article that explains decoupling simply:  
**“Decoupling”** - [Medium Article](https://medium.com/@dahnielson/decoupling-2045aadf7729)