Chapter 1. The Tar Pit

Unlike small projects, large projects face more complex management issues. It's like an animal caught in a tar pit. Large projects often get stuck in difficulties that are hard to resolve. Building a product is very different from building a single program. A product needs documentation, testing, and maintenance, which requires at least 3x the work of a single program. If you're building a system of programs that interact with each other, it costs even more, and making that system into a product multiplies the effort further. There are several pleasures in programming. There's the joy of creating something, the joy of building things that are useful to others, solving complex puzzles, working in a medium with no physical limitations, and making things that move and function. On the flip side, programming demands perfection, which is rare in other human activities. Developers often don't control their objectives, resources, or information, which are managed by others. They may also rely on poorly-designed or tested programs. And while some tasks are creative and fun, there's also a lot of work, like bug fixing.

Chapter 2. The Mythical Man-Month

More software projects fail because of lack of time than any other reason. Many things are underestimated, and management techniques that work in other fields are not always applied in software development. There are reasons for this problem: poor estimation(optimistic about how long things will take), people and time are not interchangeable(can't simply replace one person with another or expect more people to instantly make things faster), software managers don't push back enough, poor schedule tracking, and adding more people to delayed project. The more tasks there are, the more likely one will get delayed, pushing the entire project back. Something is guaranteed to go wrong when there are lots of tasks. More people doesn't mean faster work. There are two overheads: training, communication. We need proper scheduling. Time needs to be allocated for planning, coding, and debugging. If you reduce the time for testing to meet a deadline, you will likely face unexpected delays later. Unlike other engineering fields, software development often do not deal with solid data, which makes unreasonable schedules more common. If project is delayed, cut down on the tasks to meet the deadline.