Antipattern: Malignant Growth

Author(s): Felix, Andreas, Theo, Christian

Malignant growth describes the uncontrolled/badly managed growth of software systems, which leads to hardly maintainable and error prone systems.

Description

Software systems tend to grow over time. New features are added, additional applications are integrated. Applications become more complex and harder to reason about. Due to tight coupling of multiple modules or applications, changes and tests take more time to implement. Errors are more likely to be overlooked.

What are some examples?

- Big Ball of Mud
- VENOM

Why does this happen?

- A lack of planning and coordination, uncontrolled growth
- When problems are solved with temporary solutions
- Too few changes to the original architecture

When designers are faced with a choice between building something elegant from the ground up, or undermining the architecture of the existing system to quickly address a problem, architecture usually loses. Piecemeal Growth

- A lack of architecture and technology knowledge/experience

In the software world, we deploy our most skilled, experienced people early in the lifecycle. Later on, maintenance is relegated to junior staff, when resources can be scarce. The so-called maintenance phase is the part of the lifecycle in which the price of the fiction of master planning is really paid. It is maintenance programmers who are called upon to bear the burden of coping with the ever widening divergence between fixed designs and a continuously changing world. If the hypothesis that architectural insight emerges late in the lifecycle is correct, then this practice should be reconsidered. Piecemeal Growth

- Time pressure
- Low budget

How can we avoid getting into the situation in the first place?

- A certain amount of up-front planning and design is not only important, but inevitable Piecemeal Growth

- Do internal and external reviews regularly to get *feedback* about your system
- Use the feedback to steer the system's evolution
- Do changes in small, manageable chunks Piecemeal Growth
- Continuous integration
- No separate development and maintenance team
- Refactoring, as soon as needed

What are suggestions to get out of the situation if we ended up in it?

- Consolidation and refactoring
- Get an external architecture review, apply proposals for improvement
- Start from scratch