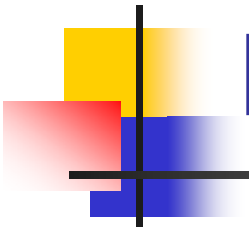
Does Code Decay? Assessing the Evidence from Change Management Data

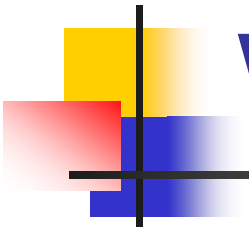
Presenter: Tao Xia

Sep 21, 2006



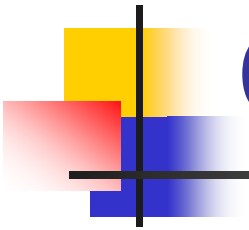
How does code decay?

- If a software system does not change, does it decay?
- Yes, because the hardware and software environment surrounding it do change
- And the requirements of the software system may change also



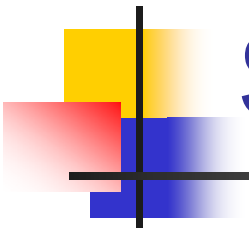
What is code decay?

- Three factors to estimate code decay
  - Cost, the resources spend on the change
  - Interval, the time requires to complete the change
  - Quality of the changed software



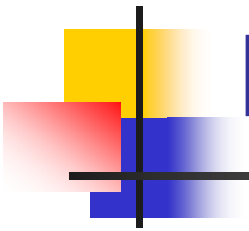
Cause of code decay

- Inappropriate architecture
- Violations of the original design principles
- Imprecise requirements
- Time pressure
- Inadequate programming tools
- Organizational environment
- Programmer variability
- Inadequate change processes



Symptoms of code decay

- Excessively complex
- Frequent changes – unstable code
- History of faults
- Widely dispersed changes
- Kludges
- Numerous interfaces (?)

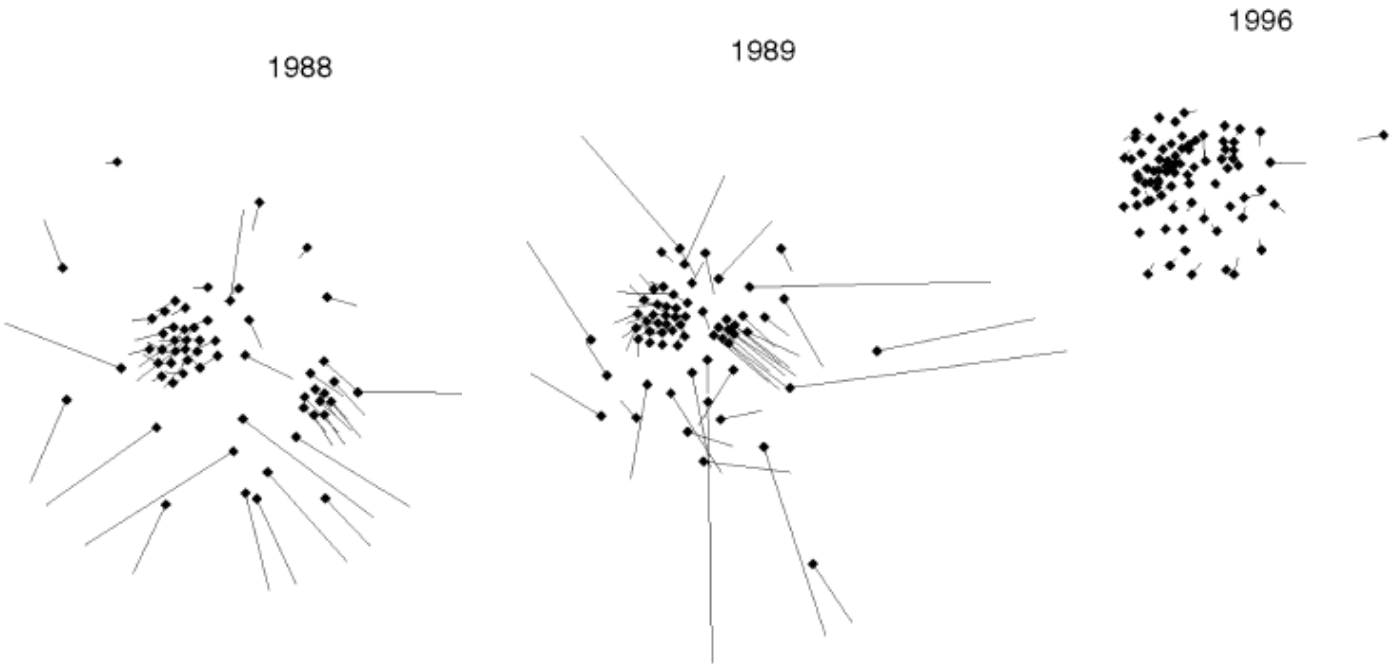


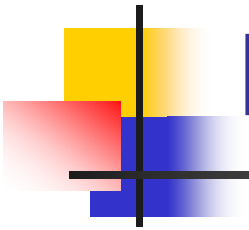
Risk factors for code decay

- Size of a module
- Age of the code
- Inherent complexity
- Organizational churn
- Ported or reused code
- Requirements load
- Inexperienced developers

Evidence of code decay (1)

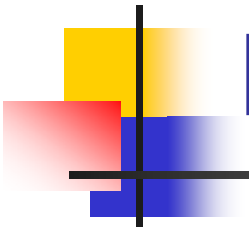
- The span of changes increases over time.
- Modularity decreases





Evidence of code decay (2)

- The number of changes to the module, the dates of these changes, and the size of changes have clear contribution to the fault rate
- Model the effort of a change from the span and size of the changes



Discussion

- The authors claim: the number of developers touching a module had no effect on its fault potential