



Software Improvement Group



Monitoring the Quality of Outsourced Software

For TOMAG 27 aug 2007

8/27/07

Arent Janszoon Ernststraat 595-H
NL-1082 LD Amsterdam
info@sig.nl
www.sig.nl

Software Improvement Group



Background

2 | 16

- Founded in 2000
- Spin-off from the Centre of Mathematics and Information Technology
- 15 year Research on Software Engineering
- Independent & no venture capital
- Winner Dutch R&D and Innovation Award 2005

Proposition



Software Improvement Group

3 | 16

- Risk analyses
- Software monitoring
- Automatic documentation of Legacy Systems

Our results and interpretations are based on measurements

The Problem



Software Improvement Group



4 | 16



Tomag 2007 - Software Monitoring - 27 aug 2007
Tobias Kuipers and Joost Visser

Software Production is “invisible”



Software Improvement Group

Even if it is done in the same building, let alone on a different continent 5 | 16

- Anything over 30 meters is too far away

Global distribution makes it even more difficult

- Language difference
- Cultural difference
- Timezone difference
- No shared sports references
- Weather
- And so on...
- (Nice article in this months Communications of the ACM)

Traditionally QA - Testing



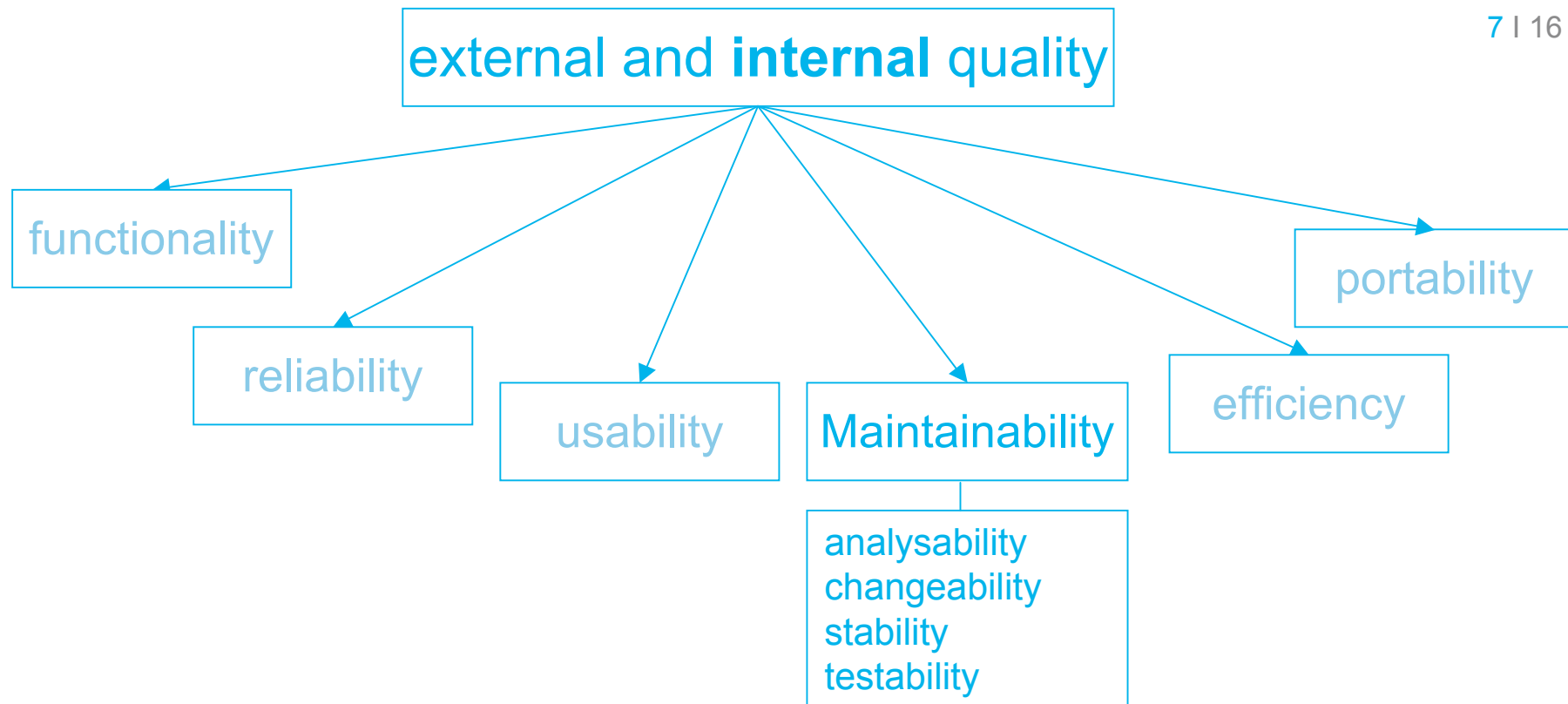
Software Improvement Group

General problem:

6 | 16

- Can you actually “test in” the quality once a project is almost finished?
- Can be done only after there is something to test
- Going back and forth between test (at client) and repair (at supplier) is time consuming
- Iterative Software Development alleviates the problem, but
- “Do these failing tests signify teething problems or a larger, structural or design flaw?”
- “Is this going to be alright, or do I kill the project right now?”

ISO/IEC 9126-1 Product quality model



Mapping source code properties onto quality sub-characteristics



Software Improvement Group

	Volume	Complexity	Unit length	Duplication	Exception handling	Aut. Unit-test quality	
Analysability	X		X	X			
Changeability		X		X			
Stability					X	X	
Testability		X	X			X	

Continuous Diagnosis



Software Improvement Group

Put a thermometer in Software Development, to get an idea of health

9 | 16

Necessarily a statistical exercise

- Because of sheer volume
- Does not matter because we are not testing, nor debugging
- We want to know the likelihood of system turning out technically sound

Aggregate from source level metrics to high level conclusion

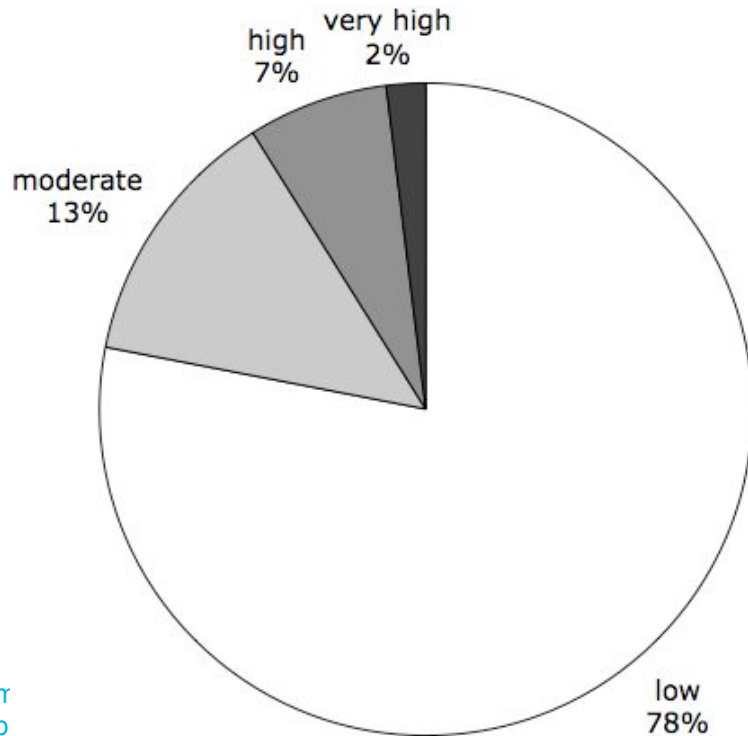
Quality profiles



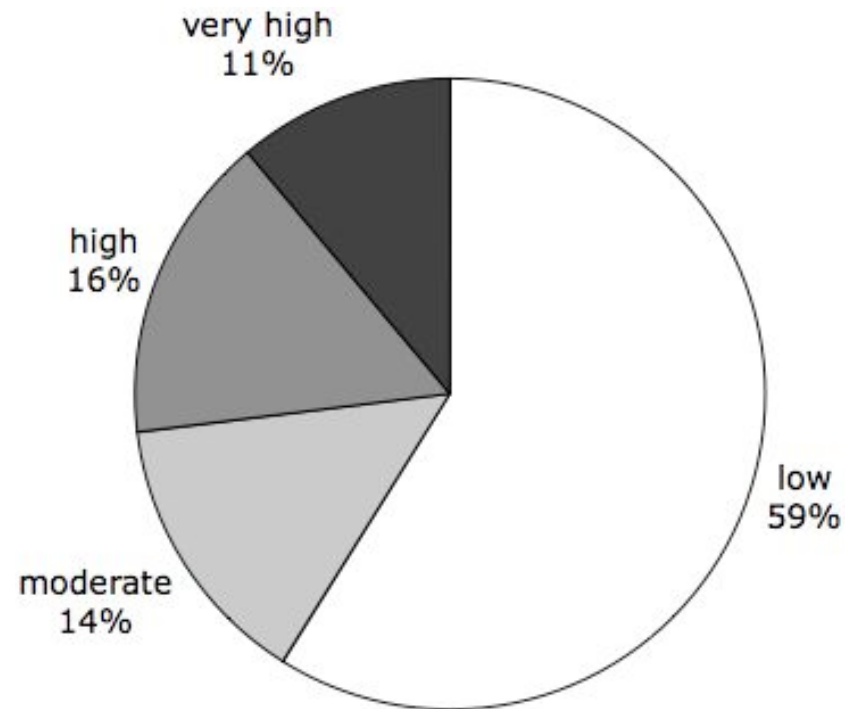
Aggregation by averaging is fundamentally flawed

10 | 16

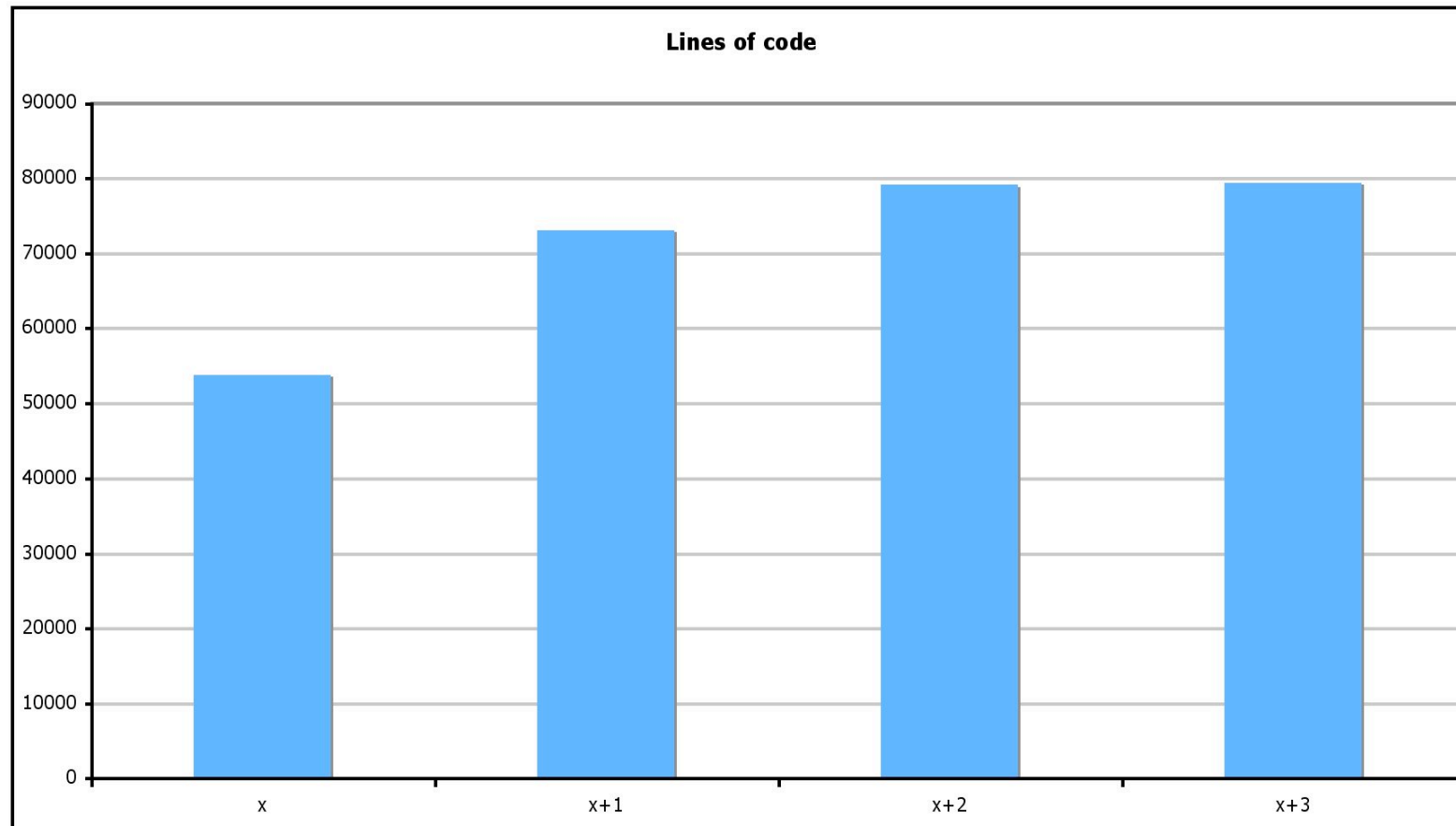
E.g. complexity



Tor
Tob



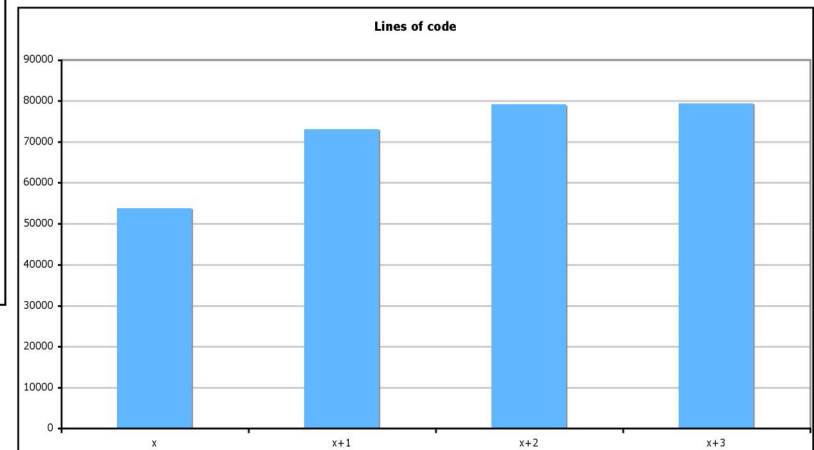
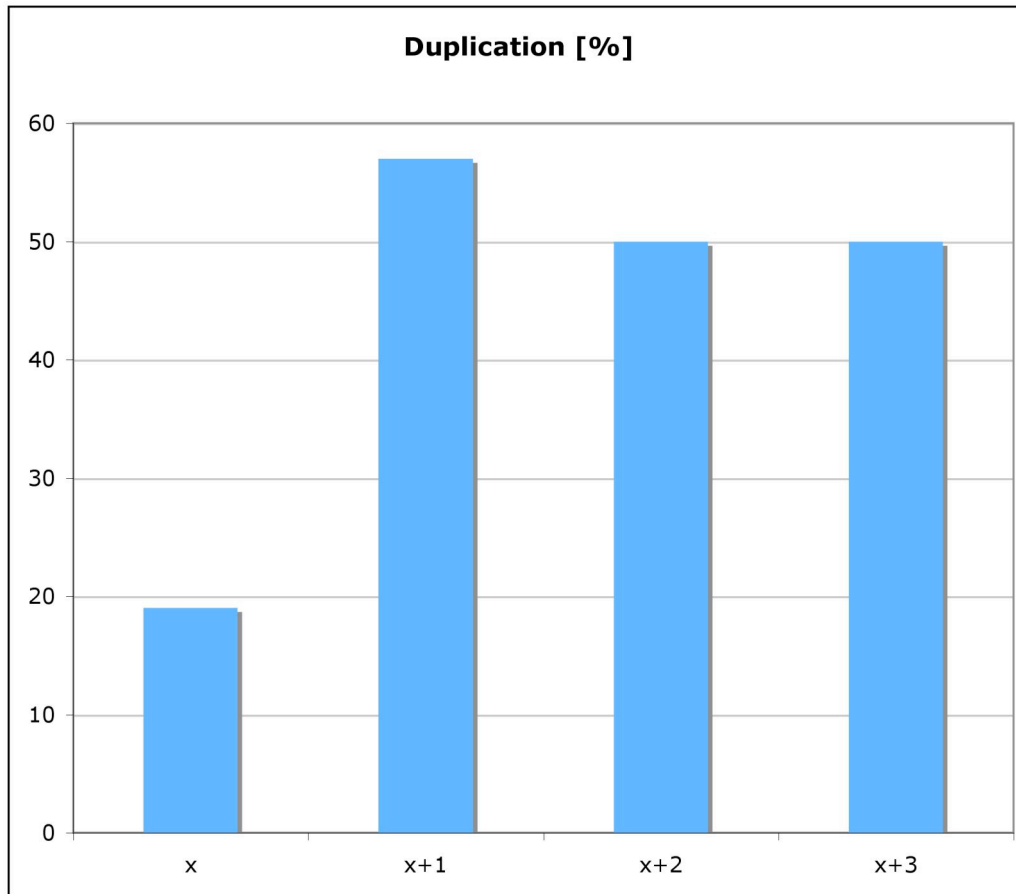
Case 1: Curbing Erosion



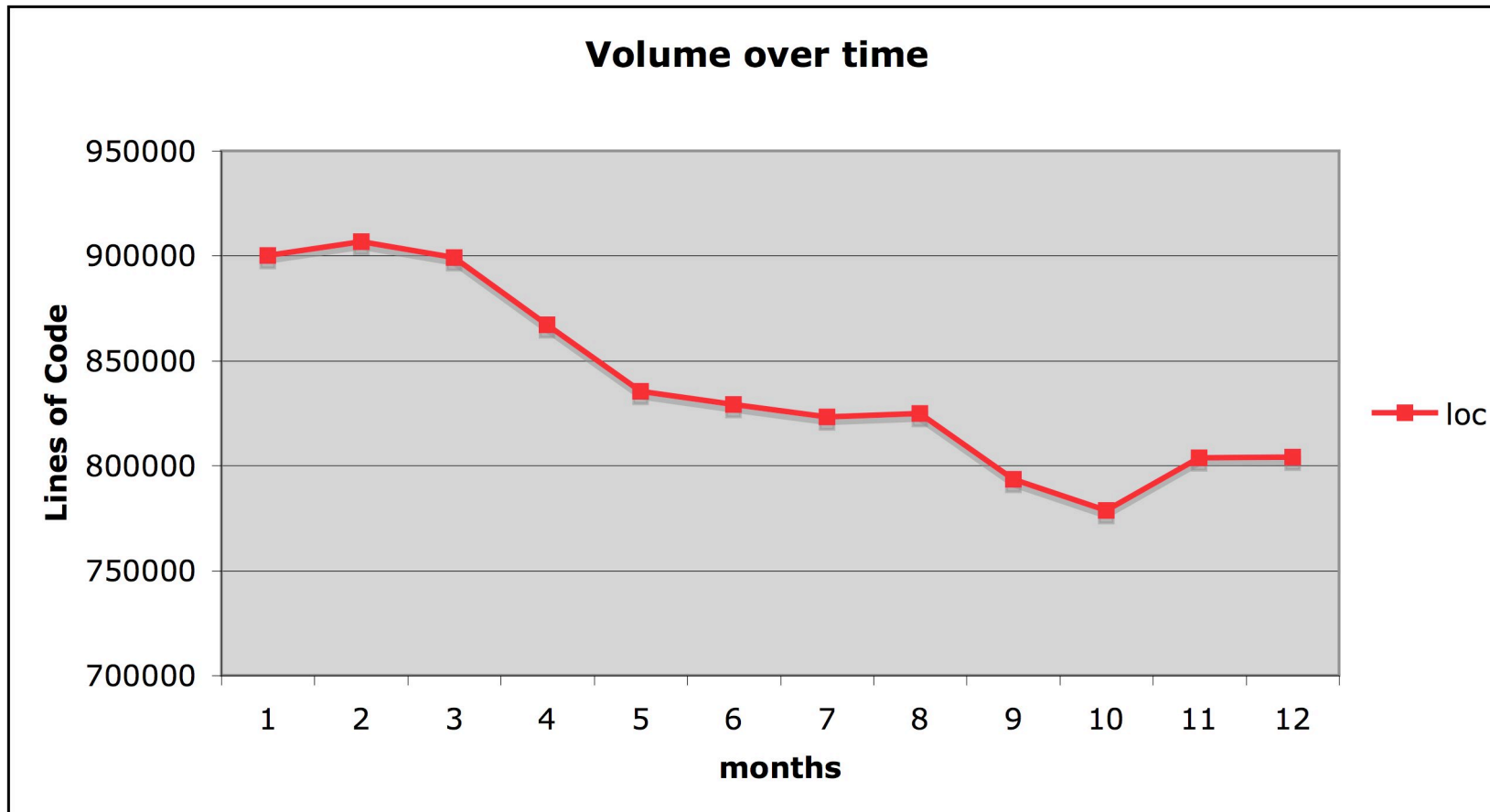
All growth is caused by duplication, there is no “real” productivity



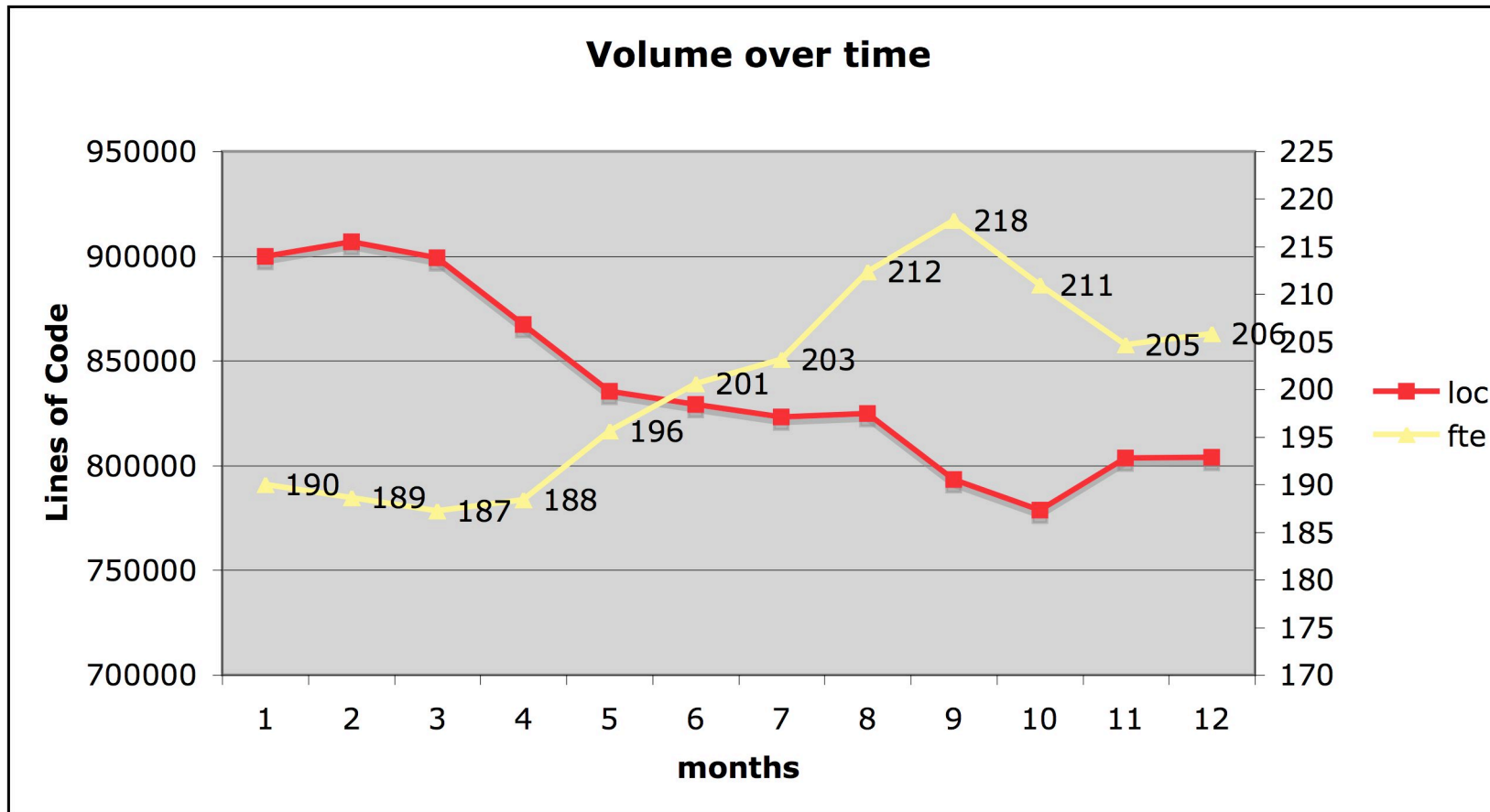
Software Improvement Group



Case 2: Systems Accounting



Case 2: How many people did you need?

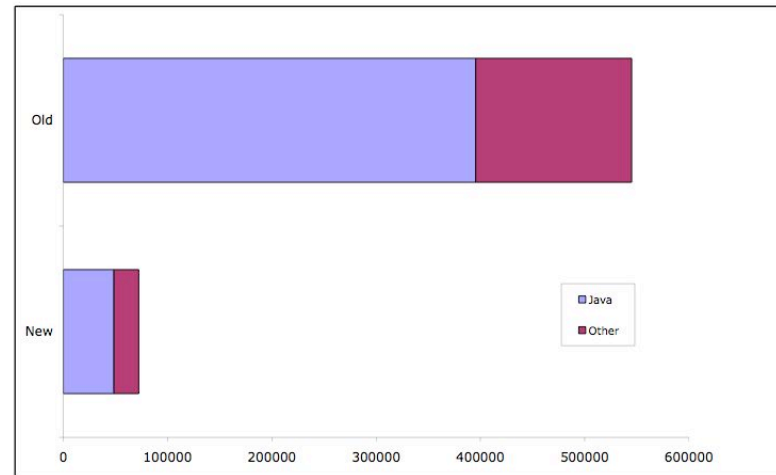
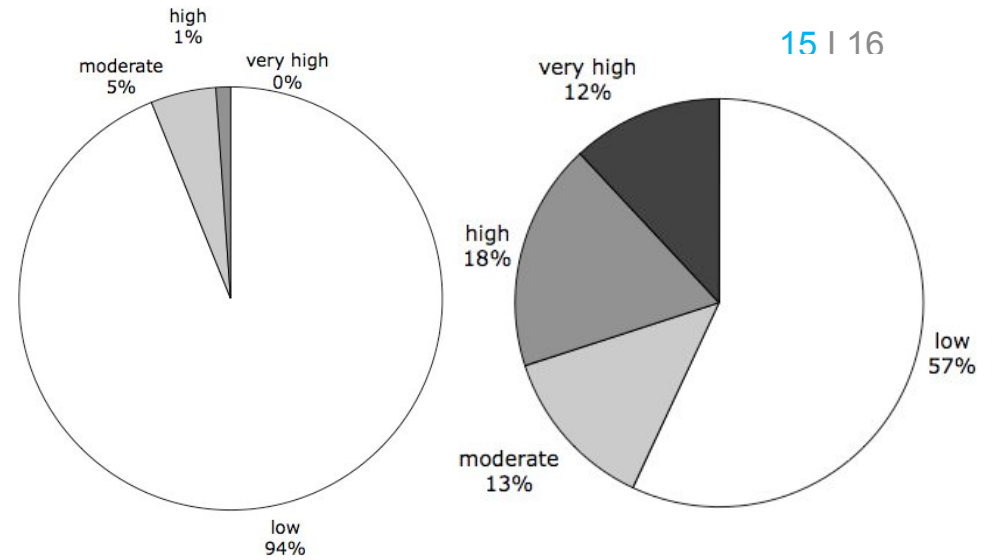
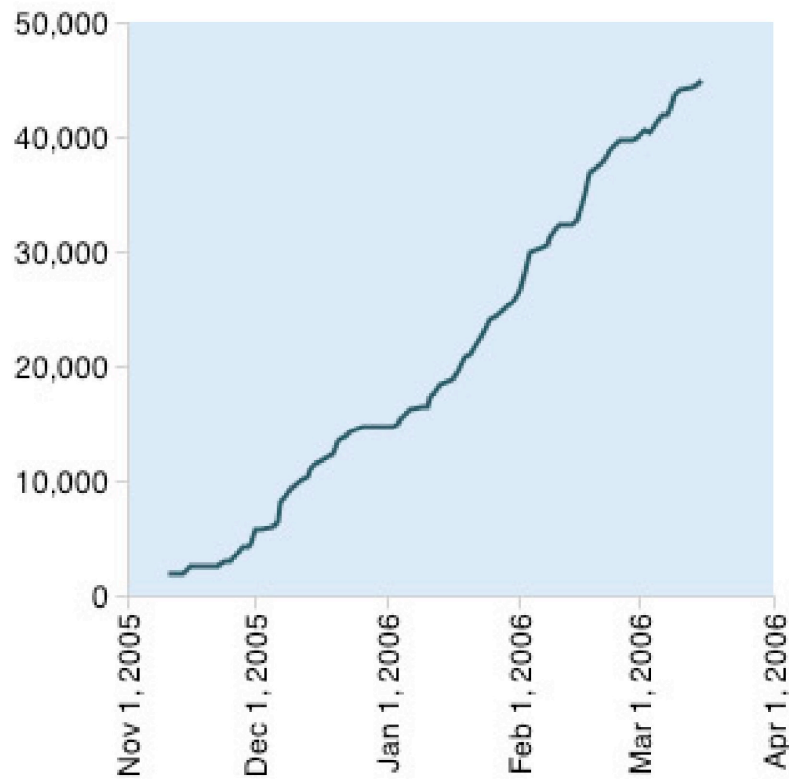




Software Improvement Group

Case 3: Learn from failure

Total lines sources



If you fell asleep, what should you remember from this presentation?



Software Improvement Group

Lessons learned

16 | 16

- Monitoring the technical quality of a software system will give you an early indicator of maintenance cost in long run
- Simple measures, but well-chosen and aggregated in meaningful ways, are effective instruments for software monitoring.
- The simultaneous use of distinct metrics can be used to zoom in on root causes of perceived quality or productivity problems
- Monitoring helps to curb system erosion in maintenance situations
- Monitoring code churn allows 'systems accounting'