— INTRODUCTION —

Lehmans Laws

Continuing Change
Used System must change to remain being useful.

Increasing Complexity
Evolving systems become more complex. Extra resources are required to keep it simple.

Software Evolution Process

- Requirements & Change management
- Change Impact Analysis
- Reengineering
- Software Modernization
- Software QA
- Quality Impact Analysis

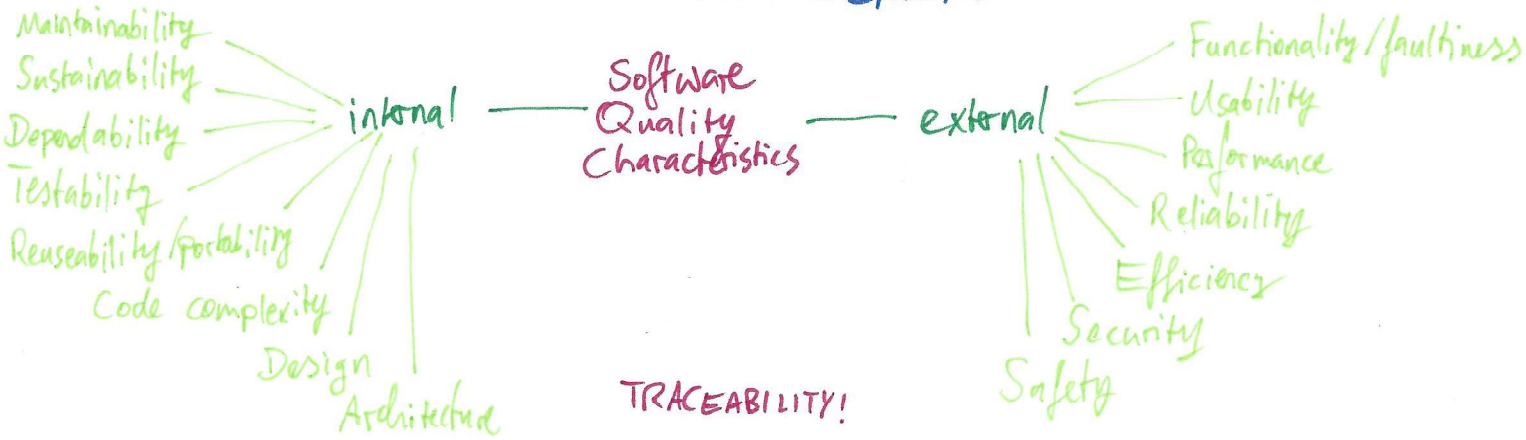
Risk Strategies

- Avoidance
- Minimization
- Contingency

Risk Categories

- Project Risks
- Product Risks
- Business Risks

— REQUIREMENTS & CHANGE MANAGEMENT —



TRACEABILITY!

— CHANGE IMPACT ANALYSIS —

Change Set
Software artifacts directly involved in change

Affected Set
Software artifacts actually affected by change

Impact Set
Software artifacts potentially affected by change.

STATIC ANALYSIS

vs

DYNAMIC ANALYSIS

CHANGE IMPACT DETERMINATION TECHNIQES

Traceability Analysis

	TC1	TC2	TC3
R1	^	0	1
R2	0	1	1
R3	0	1	0

Static Program Analysis

STATIC PROGRAM ANALYSIS

Compile-time evaluations without execution of code

Cross-Referencing

find all locations in code where program element is referenced

Data Flow Analysis

Find paths flow of data along execution

Pointer Analysis

Find possible values of pointers.
Identify pointers as unaliased, may-aliased or must-aliased.

Program Slicing

Foward Slicing: What affects variable?
Backward Slicing: What is affected by variable?

Properties of Static Program Analysis

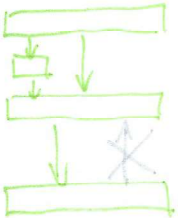
Context Sensitivity: can differentiate between different call origins
Object Sensitivity: can differentiate between different object contexts

DYNAMIC PROGRAM ANALYSIS

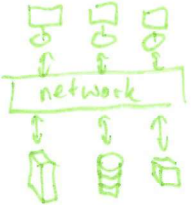
E.g. profiling, test coverage analysis

— ARCHITECTURE STYLES AND EVOLUTION —

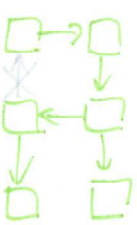
Layers



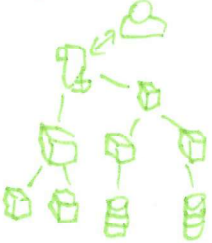
Client-Server



Pipe & Filter



Microservices



Shared Data

PAC

Presentation Abstraction Control

EVOLUTION SUPPORTING SOFTWARE DESIGN

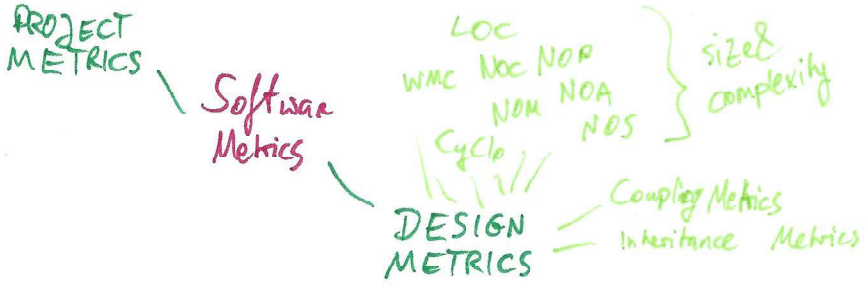
S ingle Responsibility
O pen Closed
L iskov Substitution
I nterface Segregation: Dont Depend on unused Inkfaces. Refused Bequest.
D ependency ~~Injection~~ Inversion

Stable Dependencies: Depend on components which are more stable.
Law of Demeter
Common Closure: SRP for Subsystems

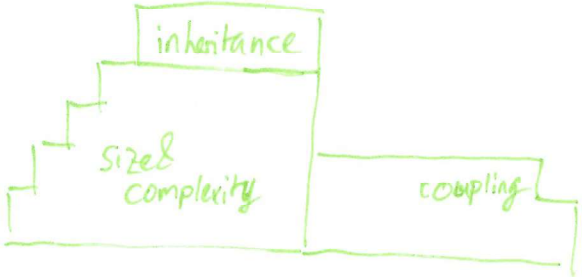
GoF Design Patterns

Creational
Structural
Behavioural

SOFTWARE MAINTAINABILITY ASSESSMENT



Overview Pyramid



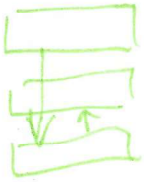
DESIGN PROBLEMS

4/5

— Maintainability Problems - Architecture —

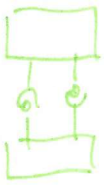
Improper Layering

super layers, layer bypass, reversed deps



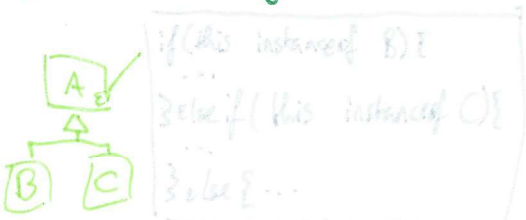
Wide Interfaces

Cyclical Dependencies

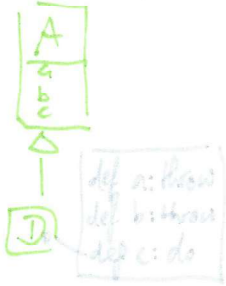


— Maintainability Problems - Design and Code —

Simulated Polymorphism



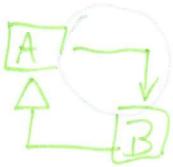
Refused Bequest



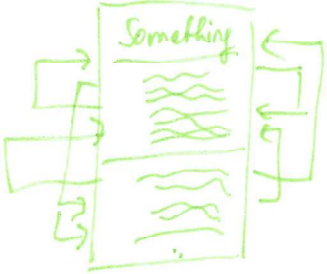
Feature Envy

much foreign data, little internal data

Knows of Derived

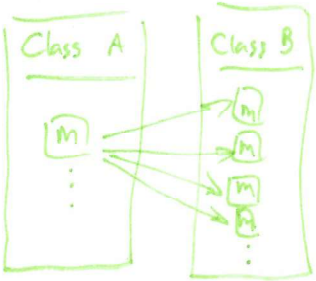


God Class

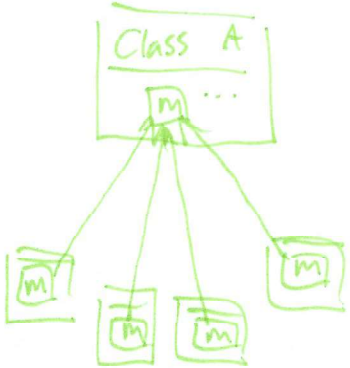


— Design Problems - Code —

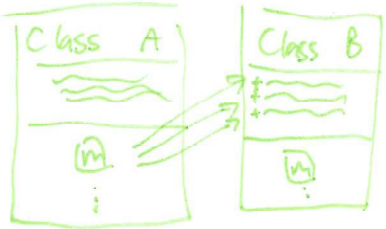
Intensive Coupling



Shotgun Surgery

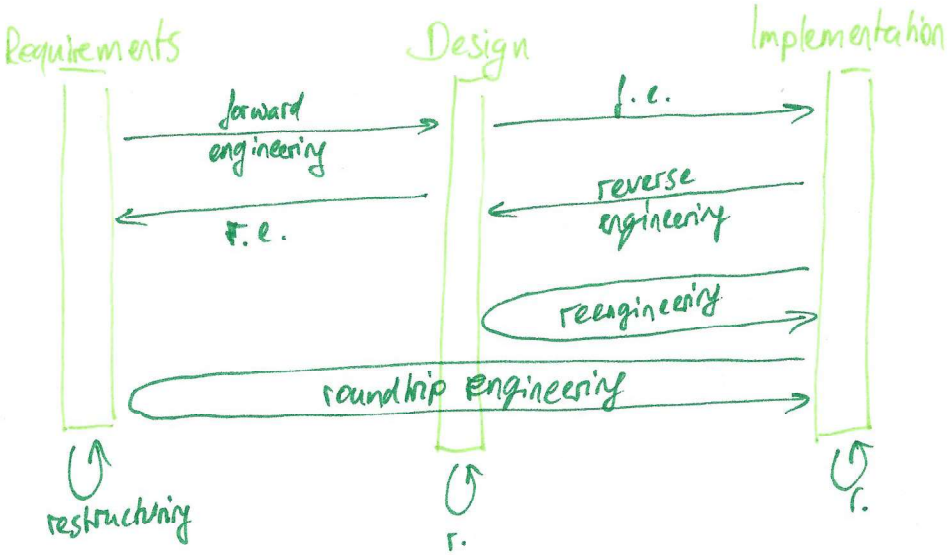


Violation of ~~Code~~ Data Encapsulation



Code Duplication

– Reengineering Legacy Systems –



– Maintainance Cost Estimation –

Top-Down Estimation
based on requirements
LOC-based, FP-based, COCOMO, by Analogy

vs

Bottom-Up-Estimation
survey of developers