Software Architecture

Laboratory 10
Modularity

Néstor Cataño
Innopolis University

Spring 2016

Modularity

Modularity

- five criteria, five rules, five principles

Five Criteria

Modularity

- Decomposability
- Composability
- Understandability
- Continuity
- Protection

Modular Decomposability

Modular Decomposability

A software construction method satisfies **Modular Decomposability** if it helps in the task of decomposing a software problem into a small number of less complex subproblems, connected by a simple structure, and independent enough to allow further work to proceed separately on each of them

Modular Composability

Modular Composability

A method satisfies **Modular Composability** if it favors the production of software elements which may then be freely combined with each other to produce new systems, possibly in an environment quite different from the one in which they were initially developed.

Modular understandability

...

A method favors **Modular Understandability** if it helps produce software in which a human reader can understand each module without having to know the others, or, at worst, by having to examine only a few of the others.

Modular Continuity

Modular Continuity

A method satisfies **Modular Continuity** if, in the software architectures that it yields, a small change in a problem specification will trigger a change of just one module, or a small number of modules.

Modular Protection

Modular Protection

A method satisfies **Modular Protection** if it yields architectures in which the effect of an abnormal condition occurring at run time in a module will remain confined to that module, or at worst will only propagate to a few neighboring modules.

Five Rules

Five Rules

- Direct Mapping
- Few Interfaces
- Small Interfaces (weak coupling)
- Explicit Interfaces
- Information Hiding

Direct Mapping

Direct Mapping

The modular structure devised in the process of building a software system should remain compatible with any modular structure devised in the process of modeling the problem domain

Few Interfaces

Few Interfaces

Weak Coupling: if two modules communicate, they should exchange as little information as possible

Small Interfaces

Small Interfaces

Every two communicating interfaces should exchange as little information as possible.

Explicit Interfaces

Explicit Interfaces

Whenever two modules **A** and **B** communicate, this must be obvious from the text of **A** or **B** or both.

Information Hiding

Information Hiding

The designer of every module must select a subset of the modules properties as the official information about the module, to be made available to authors of client modules.

Five Principles

Five Rules

- The Linguistic Modular Units principle
- The Self-Documentation principle
- The Uniform Access principle
- The Open-Closed Principle
- The Single Choice principles

Linguistic Modular Units principle

Linguistic Modular Units principle

Modules must correspond to syntactic units in the language used.

Self-Documentation

Self-Documentation

The designer of a module should strive to make all information about the module part of the module itself.

Open-Closed principle

Open-Closed principle

Modules should be both open and closed.

Open-Closed principle

Open-Closed principle

Modules should be both open and closed.

Single Choice principle

Single Choice principle

Whenever a software system must support a set of alternatives, one and only one module in the system should know their exhaustive list.

Further Reading

Object-Oriented Software Construction, 2nd Edition, by Bertrand Meyer, Chapter 3.