Software Design & Architecture
(Week-3)

Usama Musharaf

MS-CS (Software Engineering)

LECTURER *(Department of Computer Science)*

FAST-NUCES Peshawar

Agenda of Week # 3

- Software Quality Models
- Quality Attributes of a Good Design
- Design Principles
- Design Concepts

SOFTWARE QUALITY MODELS

Quality is the excellence of the product or service.

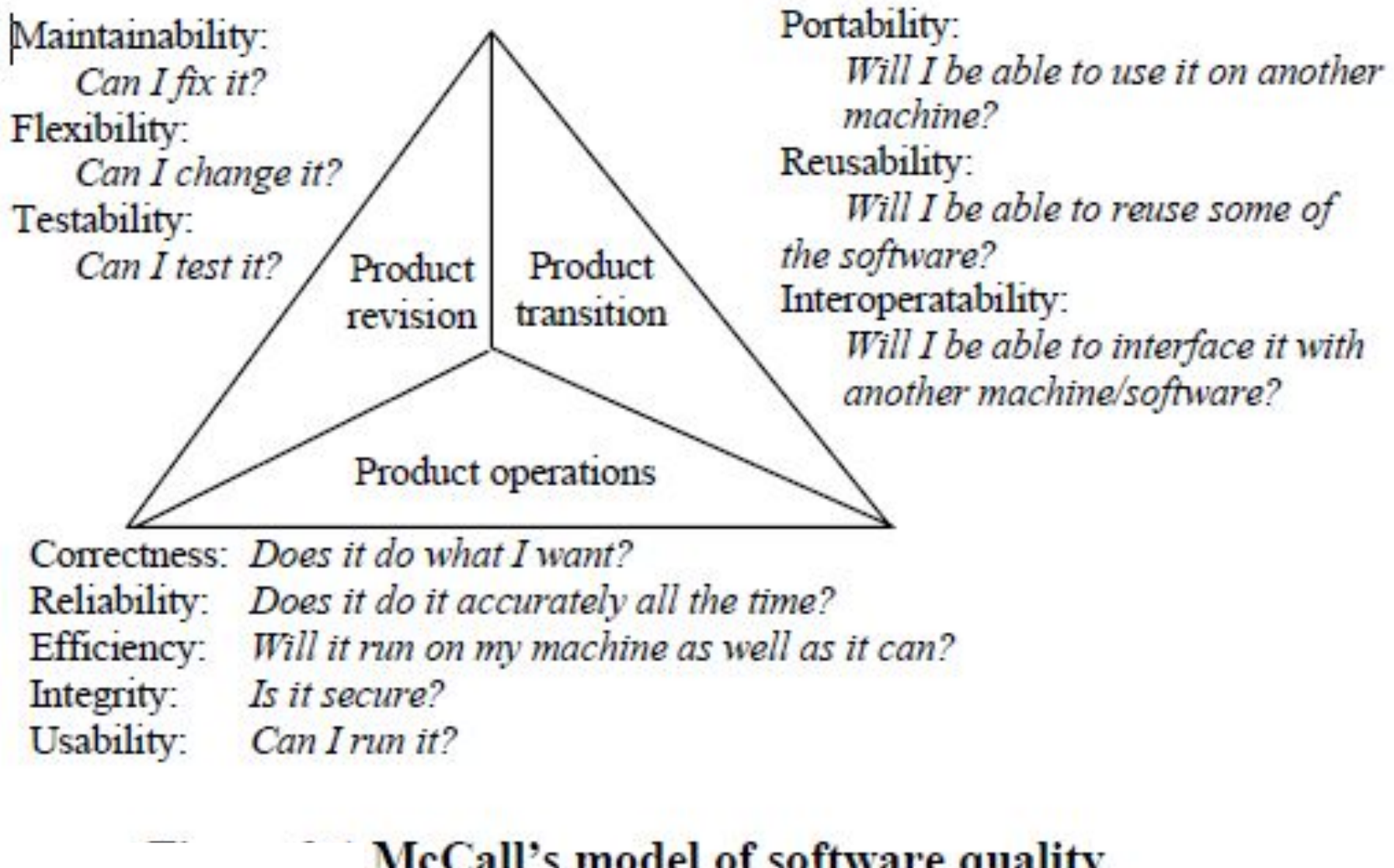
- From a user's point of view, quality is 'fitness for purpose'.
- From the manufacturing point of view, the quality of a product is the conformance to specification.

Hierarchical models

McCall divided software quality attributes into 3 groups.

- Each group represents the quality with respect to one aspect of the software system while the attributes in the group contribute to that aspect.

- Each quality attribute is defined by a question so that the quality of the software system can be assessed by answering the question.



McCall's model of software quality

Relational models

Perry’s model contains three types of relationship between the quality attributes.

- The direct relationship
- The inverse relationship
- The neutral relationship

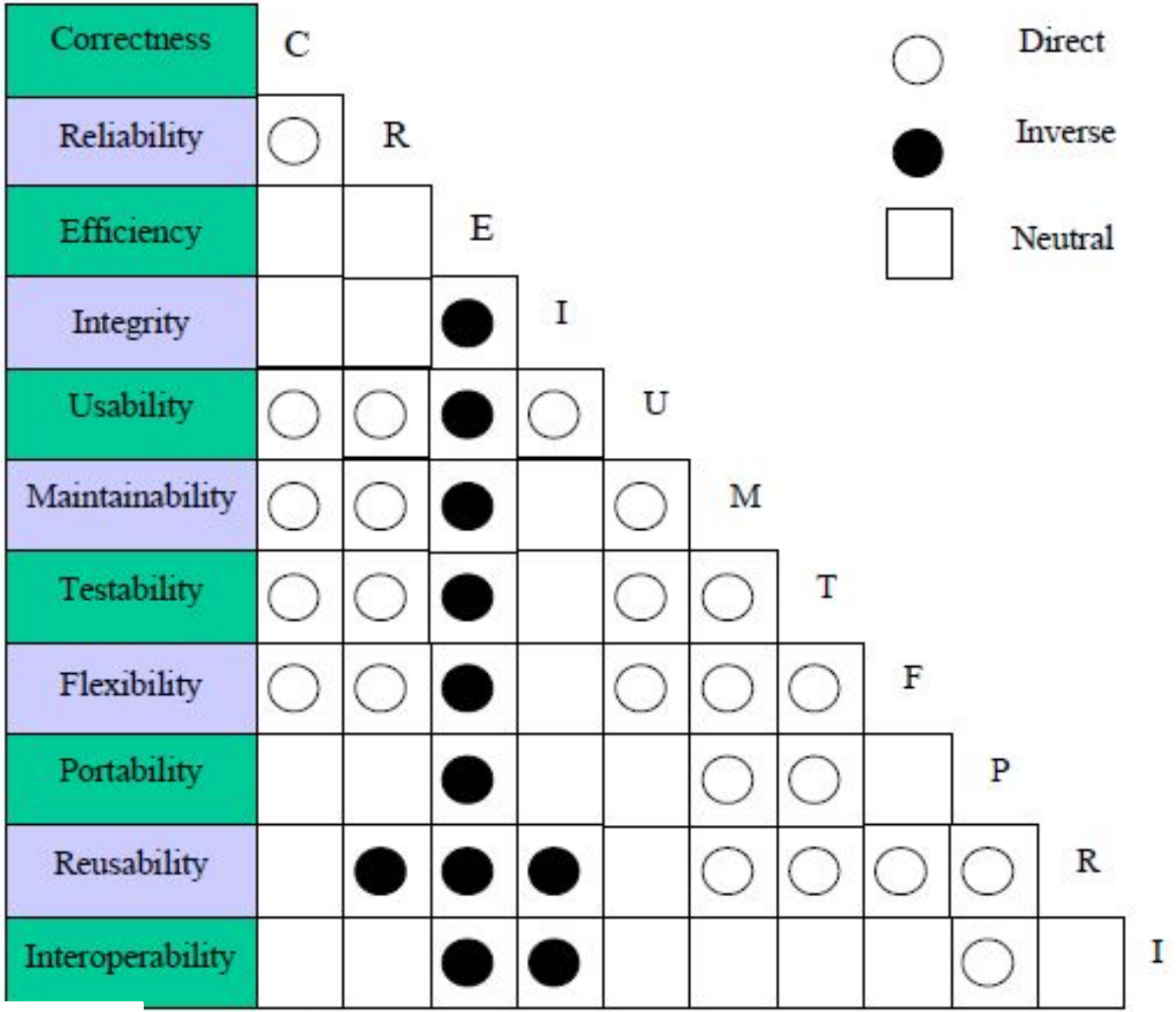


Figure 2.2 Perry's relational model of software quality

- *Integrity vs. efficiency (inverse):*

  The control of data access will need additional code, leading to a longer runtime and more storage requirement.

- *Usability vs. efficiency (inverse):*

  Improvement of HCI will need more code and data, hence the system will be less efficient.

- *Maintainability and testability vs. efficiency (inverse):*

  Compact and optimized code is not easy to maintain and test.

- *Flexibility, reusability vs. integrity (inverse):*

  Flexible data structures required for flexible and reusable software increase the data security problem.

- *Reusability vs. maintainability (direct):*

- *Portability vs. reusability (direct):*

  Portable code is likely to be easily used in other environments. The code is likely well-structured and easier to be reused.

- *Correctness vs. efficiency (neutral):*

  The correctness of code has no relation with its efficiency. Correct code may be efficient or inefficient in operation.

Design Principles

1- The design process should not suffer from “tunnel vision.”

A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job, and the design concepts.

2- The design should be traceable to the analysis model.

Design Principles

3- The design should not reinvent the wheel.

- Systems are constructed using a set of design patterns.
- These patterns should always be chosen as an alternative to reinvention.
- Time is short and resources are limited!

Design Principles

4- The design should “minimize the intellectual distance” between the software and the problem as it exists in the real world.

5- The design should exhibit uniformity and integration

Design Principles

6- The design should be structured to accommodate change

7- The design should be assessed for quality as it is being created

Design Concepts

Fundamental Concepts of Design

- **abstraction**—data, procedure, control
- **refinement**—elaboration of detail for all abstractions
- **modularity**—compartmentalization of data and function
- **architecture**—overall structure of the software
  - Styles and patterns
- **procedure**—the algorithms that achieve function
- **hiding**—controlled interfaces

Abstraction

"Capture only those details about an object that are relevant to current perspective"

Suppose we want to implement abstraction for the following statement,

"Ali is a PhD student and teaches BS students"

*Here object Ali has two **perspectives** one is his **student perspective** and second is his **teacher perspective.***

Abstraction

We can sum up Ali's attributes as follows,

Attributes	Behaviour
Name	Study
Age	DevelopExam
Student Roll No	GiveExam
Year of Study	TakeExam
CGPA	PlaySports
Employee ID	Eat
Designation	DeliverLecture
Salary	Walk

Abstraction

A cat can be viewed with different perspectives.

Ordinary Perspective	Surgeon's Perspective
A pet animal with	A being with
Four Legs	A Skeleton
A Tail	Heart
Two Ears	Kidney
Sharp Teeth	Stomach

Abstraction

A car can be viewed with different perspectives.

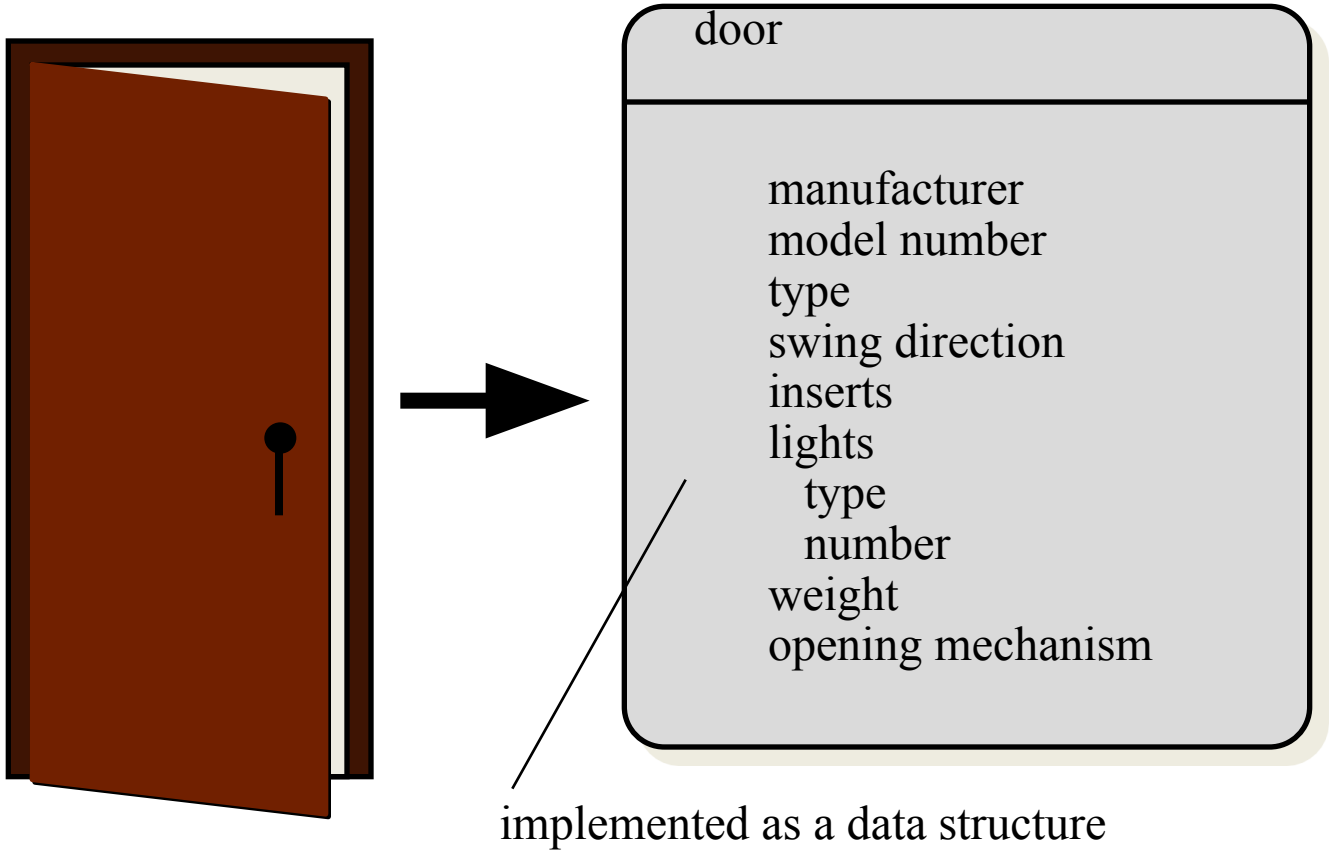
Driver's View

Engineer's View

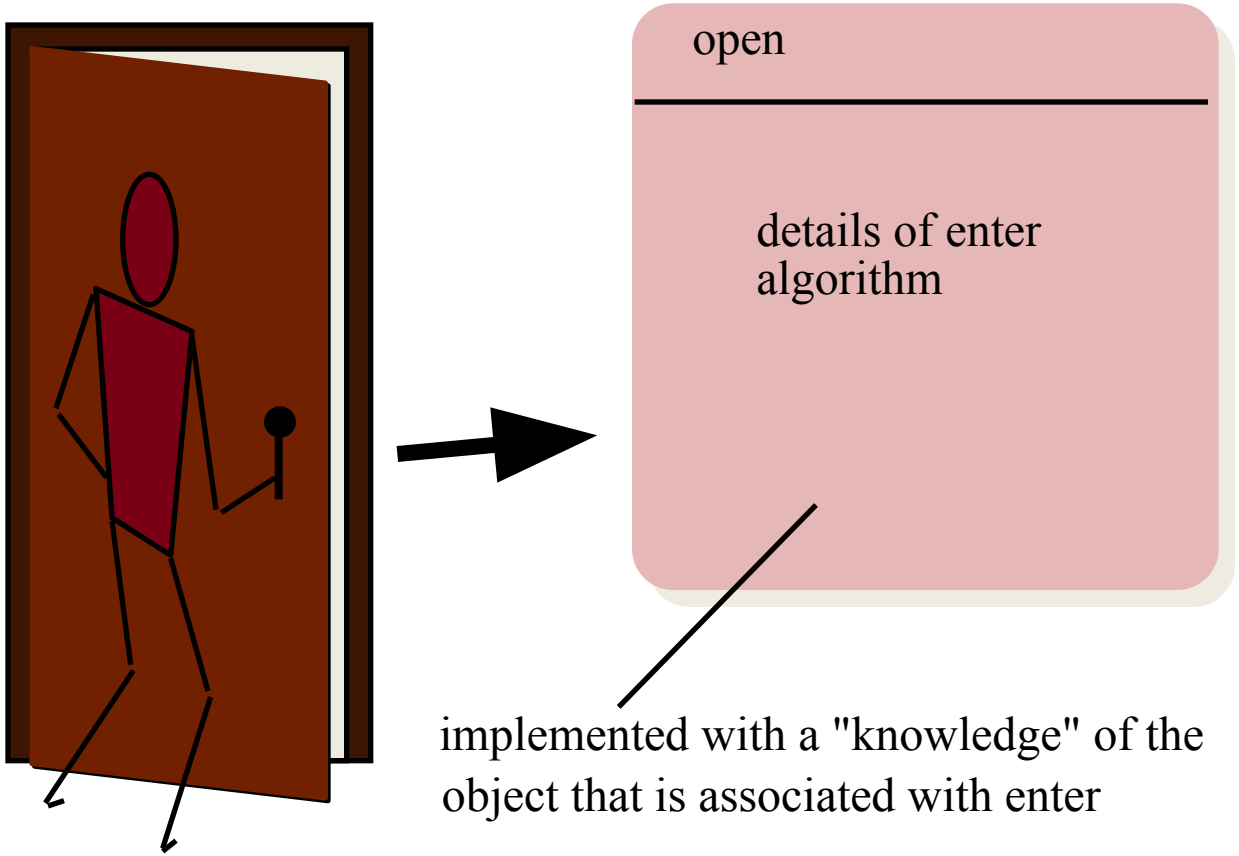
Abstraction Advantages

- It helps us understanding and solving a problem using object oriented approach as it hides extra irrelevant details of objects.
- Focusing on single perspective of an object provides us freedom to change implementation for other aspects of for an object later.
- *Abstraction is used for achieving information hiding as we show only relevant details to related objects, and hide other details.*

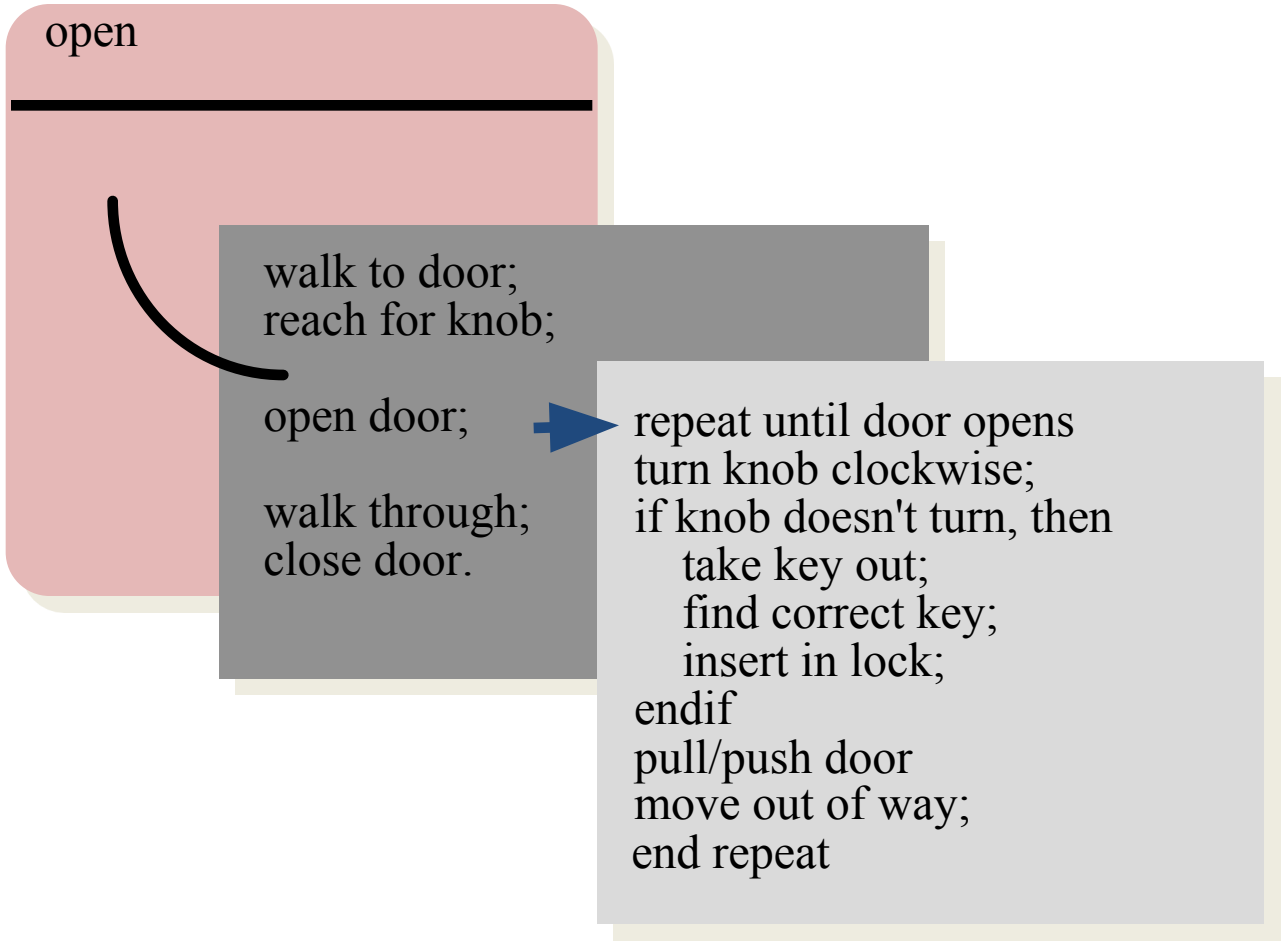
Data Abstraction



Procedural Abstraction

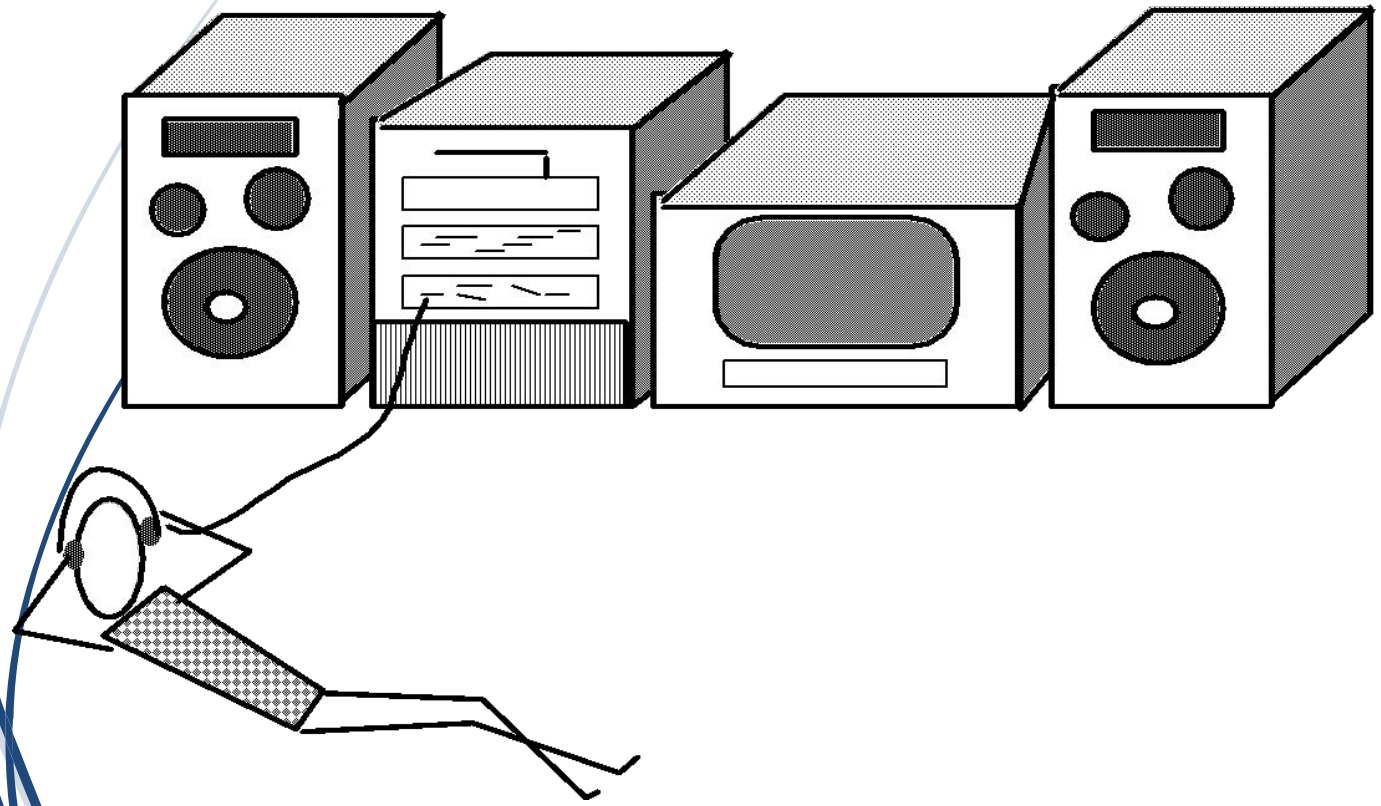


Stepwise Refinement



Modular Design

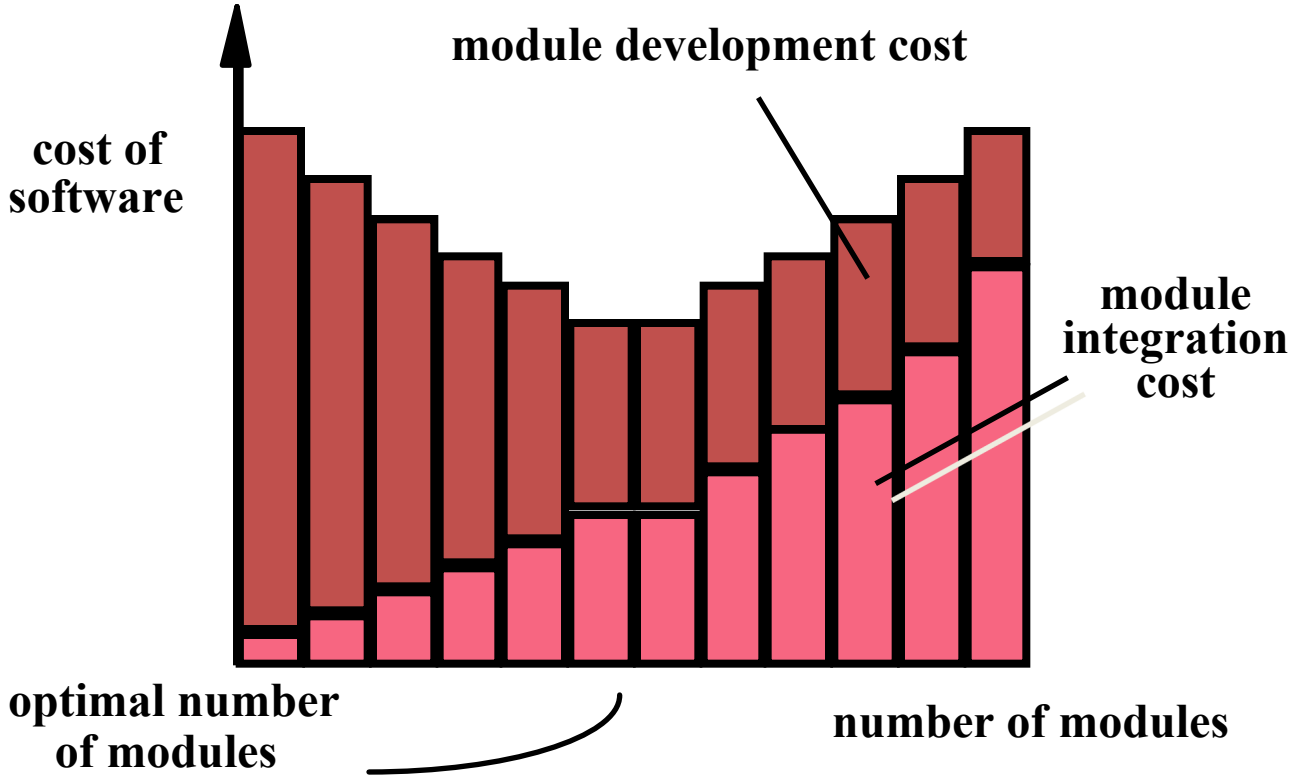
easier to build, easier to change, easier to fix …



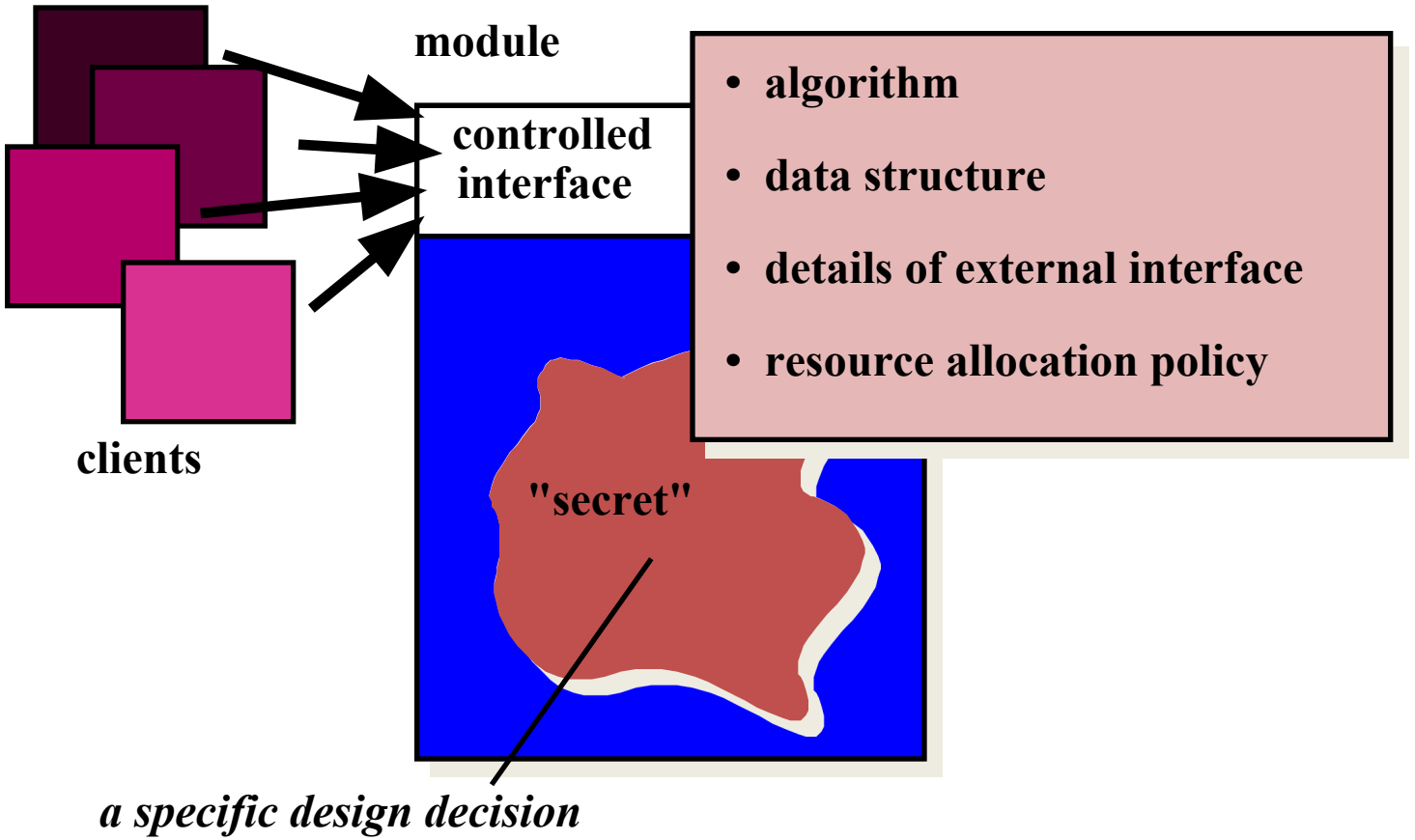
- Easier to manage
- Easier to understand
- Reduces complexity
- Delegation / division of work
- Fault isolation
- Independent development
- Separation of concerns
- Reuse

Modularity: Trade-offs

What is the "right" number of modules for a specific software design?



Information Hiding



Information Hiding

- Design the modules in such a way that information (data & procedures) contained in one module is inaccessible to other modules that have no need for such information.

- Independent modules.
- Benefits:
  - when modifications are required, it reduces the chances of propagating to other modules.

EFFECTIVE MODULAR DESIGN

Functional Independence

COHESION - the degree to which a module performs one and only one function.

COUPLING - the degree to which a module is "connected" to other modules in the system.

Coupling

Coupling is a measure of independence of a module or component.

Loose coupling means that different system components have loose or less reliance upon each other.

Hence, changes in one component would have a limited affect on other components.

Coupling

High coupling causes problems

- Change propagation- ripple effect
- Difficulty in understanding
- Difficult reuse

Cohesion

Cohesion is a measure of the degree to which the elements of the module are functionally related.

It is the degree to which all elements directed towards performing a single task are contained in the component.

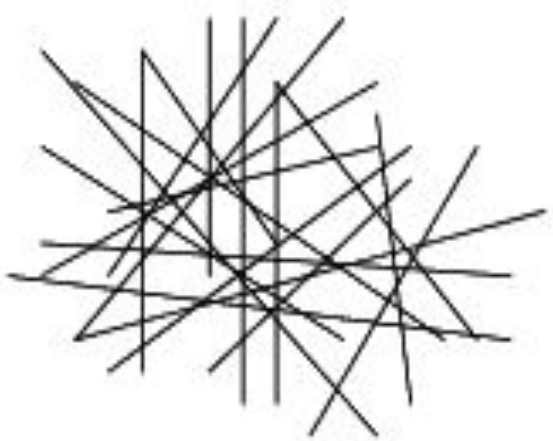
Basically, cohesion is the internal glue that keeps the module together.

A good software design will have high cohesion

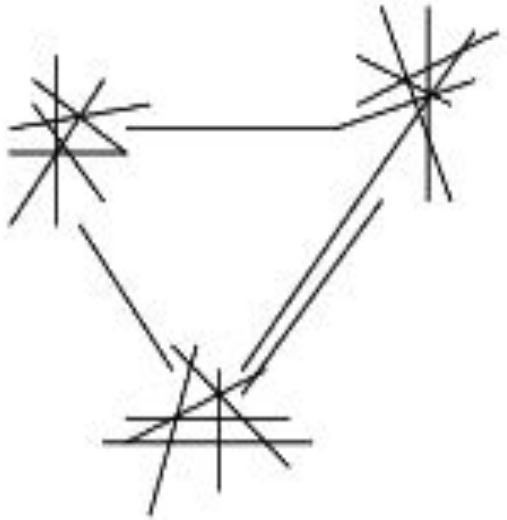
Coupling & Cohesion

A Software should be Lesly coupled and highly cohesive.

Coupling

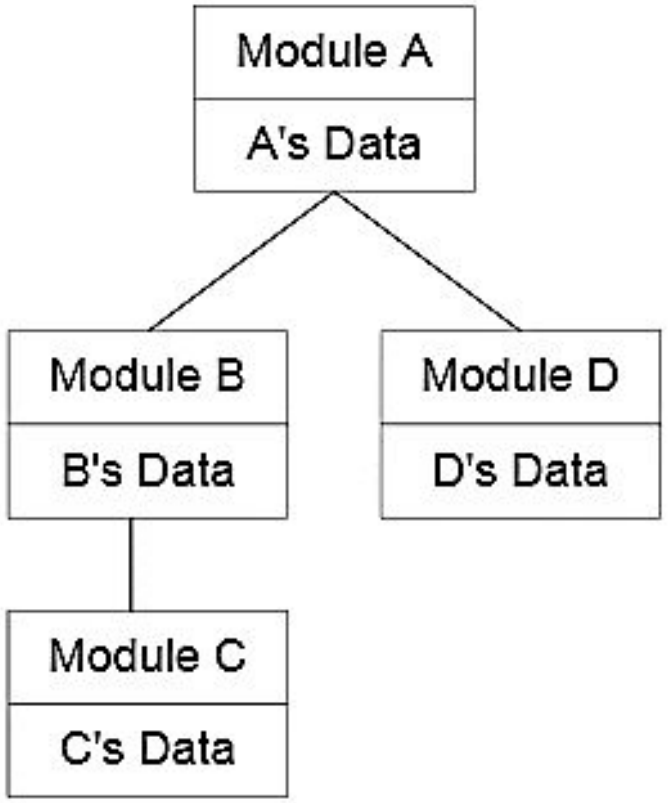


High Coupling

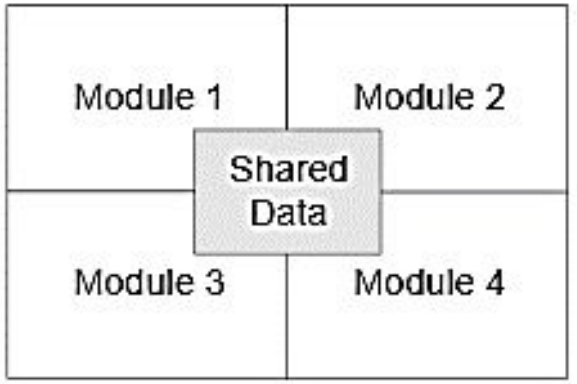


Low Coupling

Coupling



Low Coupling



High Coupling

HAVE A GOOD DAY !