COMP90041
Programming and Software Development

Modular Design

Topics covered

- **Modularity**
- **Modular Design Criteria**
- **Modular Design Rules**

Modularity

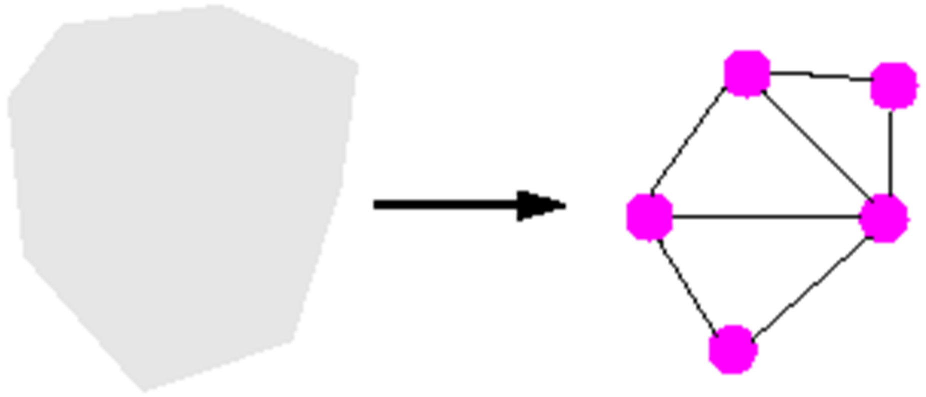
- To manage complexity and improve quality.
- *Module*: The basic unit of decomposition of our systems.
- *Modular Design/Programming*: Designing /Constructing software based on modules.

Modular Design Criteria

1. Decomposability.
2. Composability.
3. Understandability.
4. Continuity.
5. Protection.

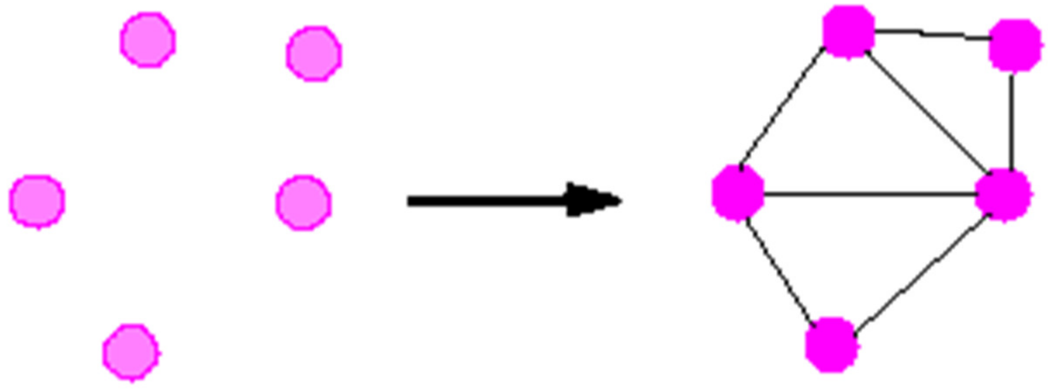
1. Modular Decomposability

A *modular design* method facilitates the task of decomposing a problem into a small number of less complex sub-problems, connected by a simple structure, and independent enough to allow further work to proceed separately on each of them.



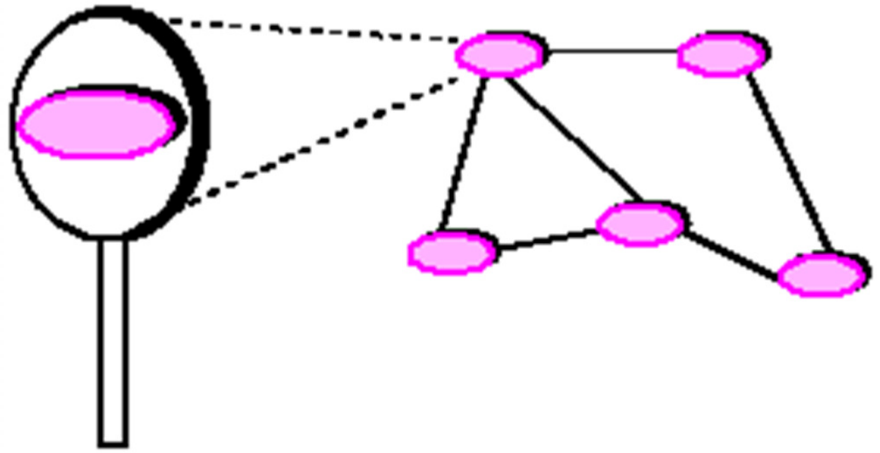
2. Modular Composability

A modular design method facilitates the production of software components which may then be freely combined with each other to produce new systems.



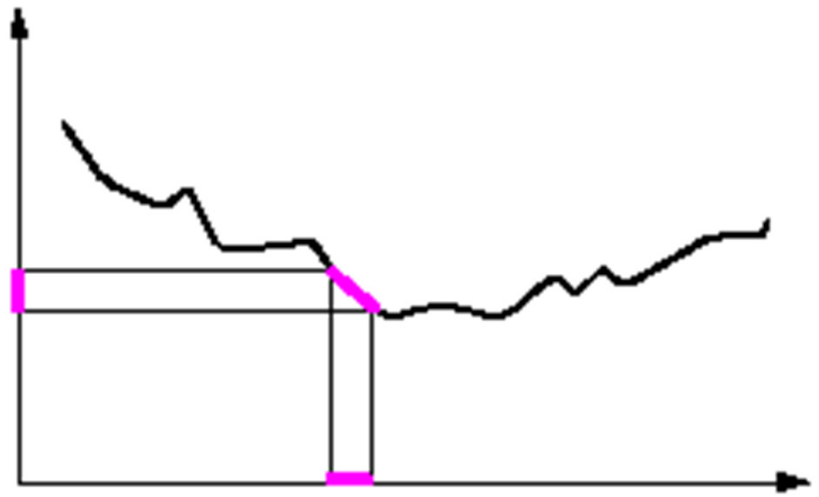
3. Modular Understandability

A design method that 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 them.



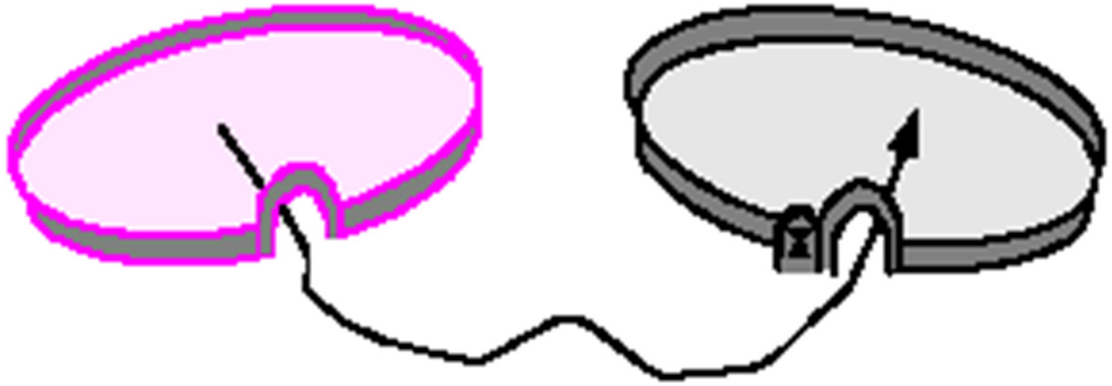
4. Modular Continuity

A design method satisfies this criterion if, a small change in a problem specification will trigger a change of just one module, or a small number of modules.



5. Modular Protection

A design method satisfies this criterion 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.



Modular Design Rules

1. Direct Mapping
2. Few Interfaces
3. Small interfaces
4. Explicit Interfaces
5. Information Hiding

1. Direct Mapping

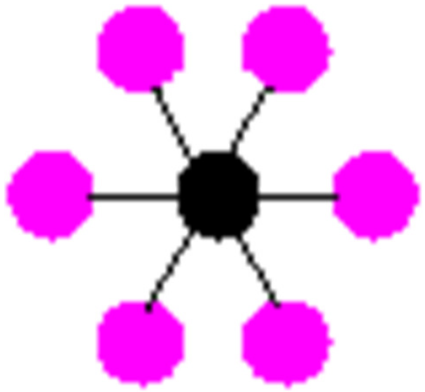
The modular structure devised in the process of building a software system should remain compatible with any modular structure devised in the process of modeling the problem domain.

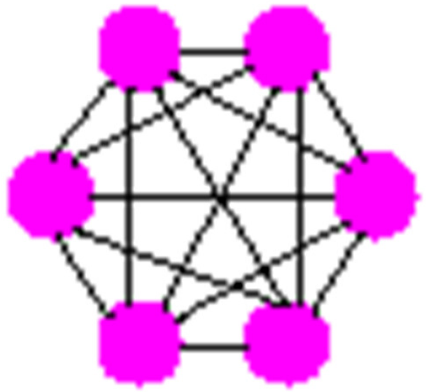
Modular Criteria: Continuity, Decomposability

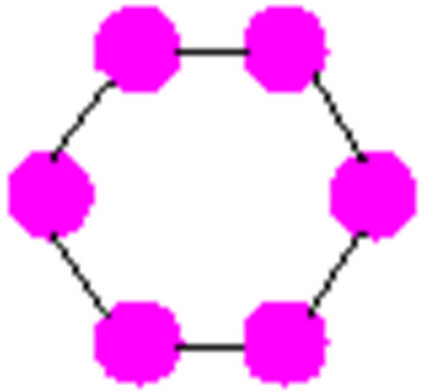
2. Few Interfaces

Every module should communicate with as few others as possible.

Modular Criteria: Continuity, Protection, Understandability, Composability and Decomposability


(A)

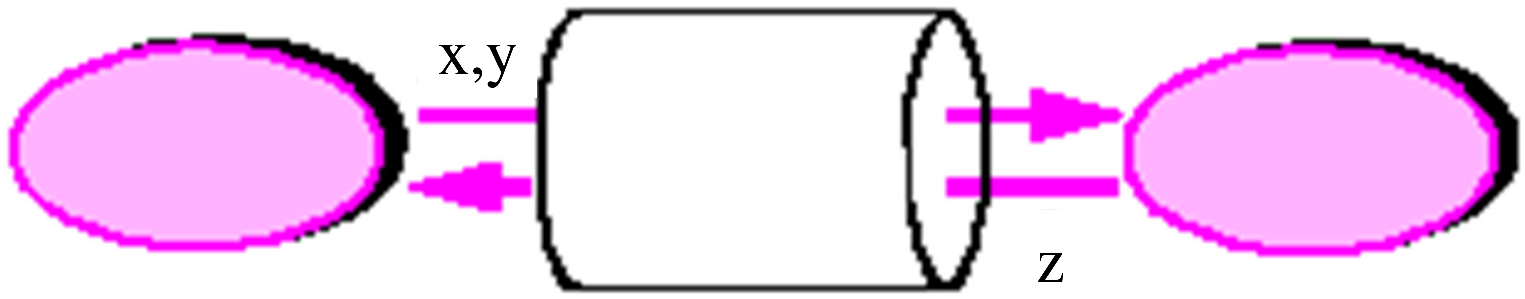

(B)


(C)

3. Small Interfaces

If two modules communicate, they should exchange as little information as possible.

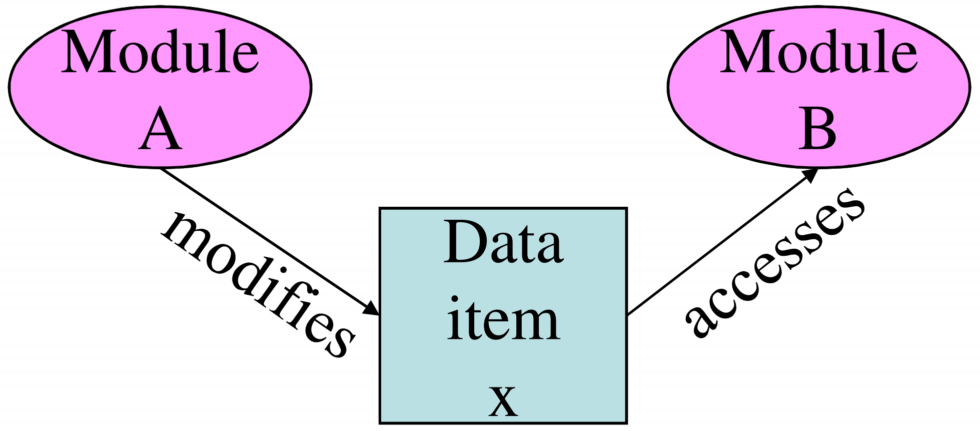
Modular Criteria: Continuity, Protection



4. Explicit Interfaces

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

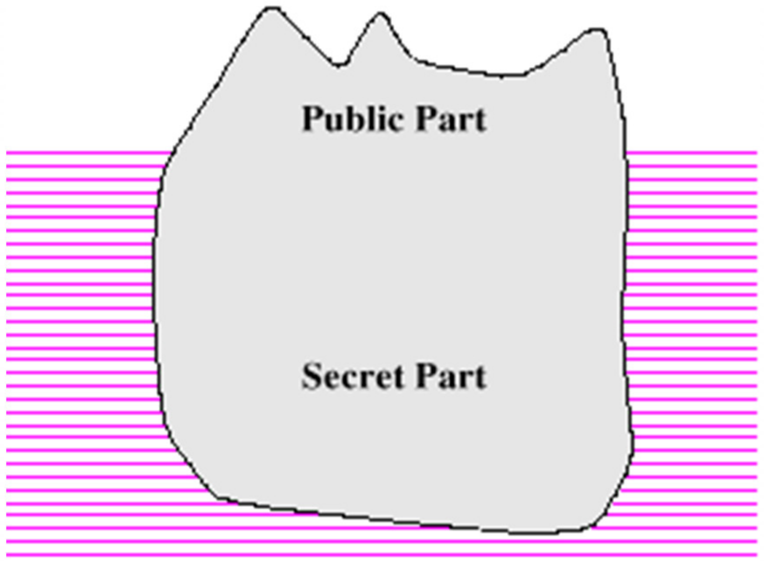
Modular Criteria: Decomposability, Composability, Continuity and Understandability.



5. Information Hiding

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

Modular Criteria: Continuity



Abstract Data Type (ADT)

- A structure that contains both data and actions to be performed on that data.
  - Meets all modularity criteria.
  - Can follow all modularity rules.

Summery

- Modularity is the way to go!
- There are criteria to conform to and rules to follow
- ADT is a good module.

Software Engineering References:

- Pfleeger, S. and Atlee, J. *Software Engineering: Theory and Practice*, Pearson-Prentice Hall, Third Edition, 2006
- Object-oriented software construction, by B. Meyer, 2nd Edition, Prentice Hall, 1997.
- Sommerville I., *Software Engineering*, 7th ed., Addison-Wesley, 2004.
- Pressman, R.S., *Software Engineering: A Practitioner's Approach*, 5th ed., McGraw-Hill, 2001.
- Van Vliet, H., *Software Engineering*, 2nd ed., John Wiley and Sons, 2000.
- Arlow J. and Neustadt I., *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison-Wesley, 2001.