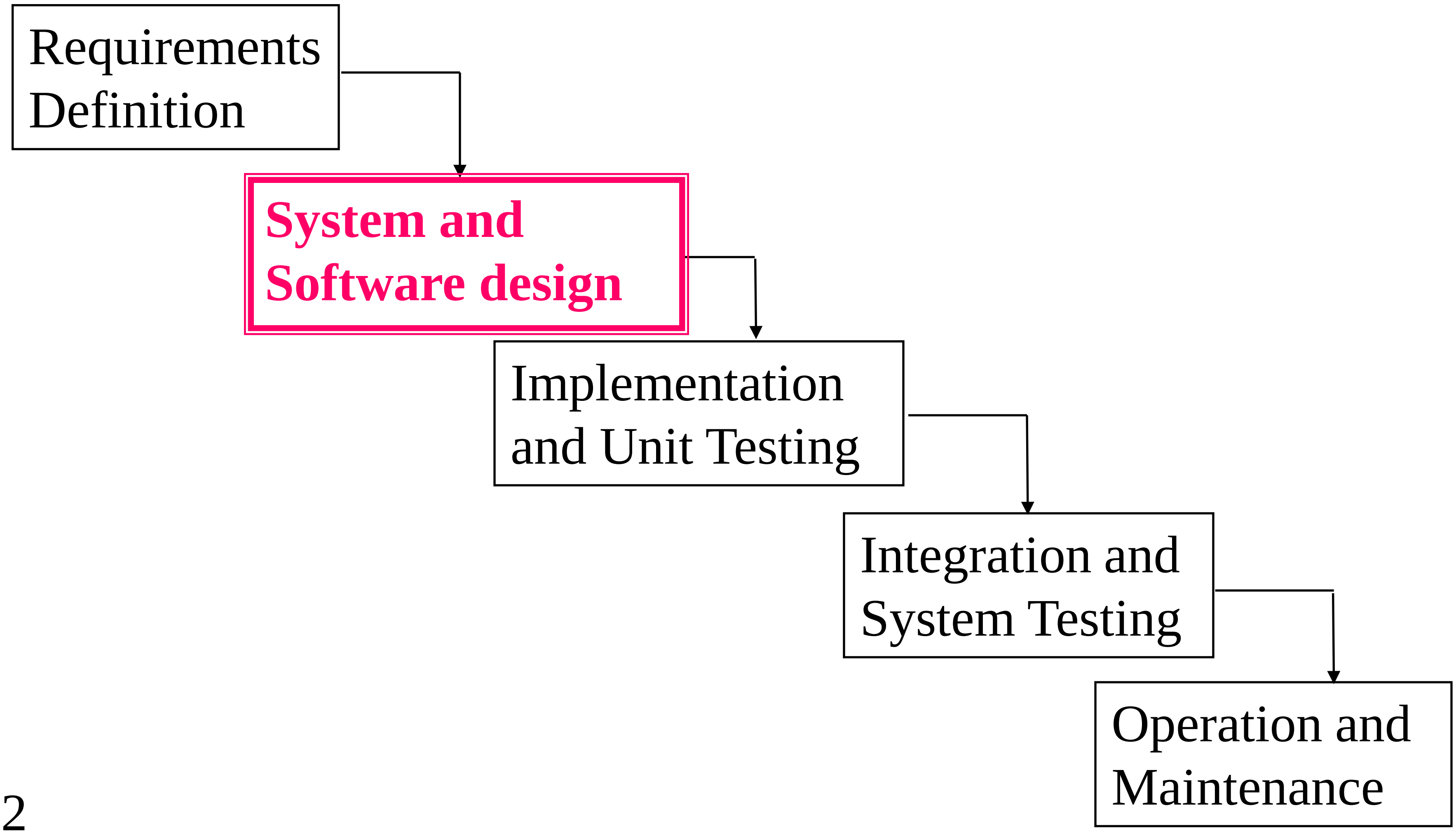
Object-Oriented Design I

The Waterfall Model



Requirements Definition

System and Software design

Implementation and Unit Testing

Integration and System Testing

Operation and Maintenance

Useful Texts

Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language*. Addison-Wesley 1999.

Grady Booch, *Object-Oriented Analysis and Design with Applications*, second edition. Benjamin/Cummings 1994.

Rob Pooley, Perdita Stevens, *Using UML Software Engineering with Objects and Components*. Addison-Wesley 1999.

The Importance of Modeling

- A model is a simplification of reality.
- We build models so that we can better understand the system we are developing.
- We build models of complex system because we cannot comprehend such a system in its entirety.

Models can be informal or formal. The more complex the project the more valuable a formal model becomes.

BRJ

Principles of Modeling

- The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
- Every model can be expressed at different levels of precision.
- The best models are connected to reality.
- No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models.

BRJ

The Unified Modeling Language

UML is a standard language for modeling software systems.

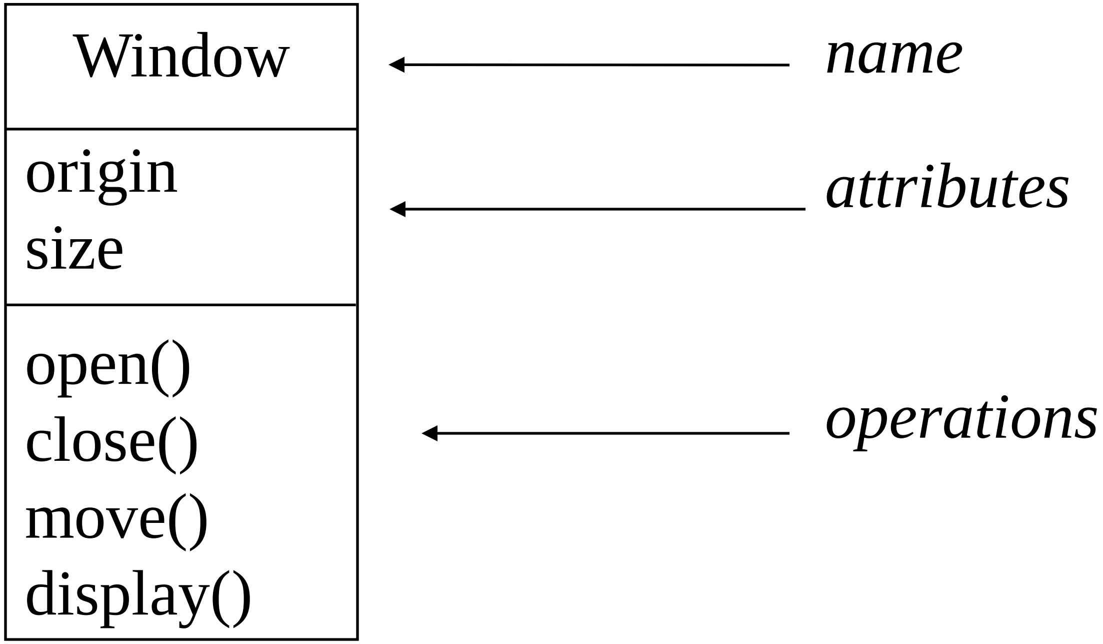
- Serves as a bridge between the requirements specification and the implementation.

- Provides a means to specify and document the design of a software system.

- Is process and programming language independent.

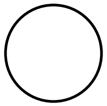
- Is particularly suited to object-oriented program development.

Notation: Classes



A **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

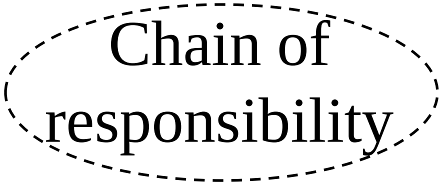
Notation: Interface



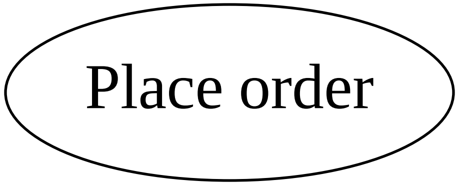
ISpelling

An **interface** is a collection of operations that specify a service of a class or component, i.e., the externally visible behavior of that element.

Notation: Collaboration & Use Case

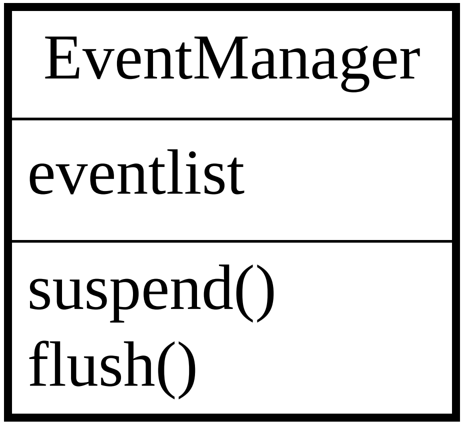

Chain of responsibility

A **collaboration** defines an interaction, i.e., a society of roles and other elements that work together to provide some cooperative behavior.


Place order

A **use case** is a description of a set of sequence of actions that a system performs that yields an observable result.

Notation: Active Class


EventManager

eventlist

suspend()
flush()

An **active class** is a class whose objects own one or more processes or threads and therefore can initiate control activity.