CS350
SE Principles

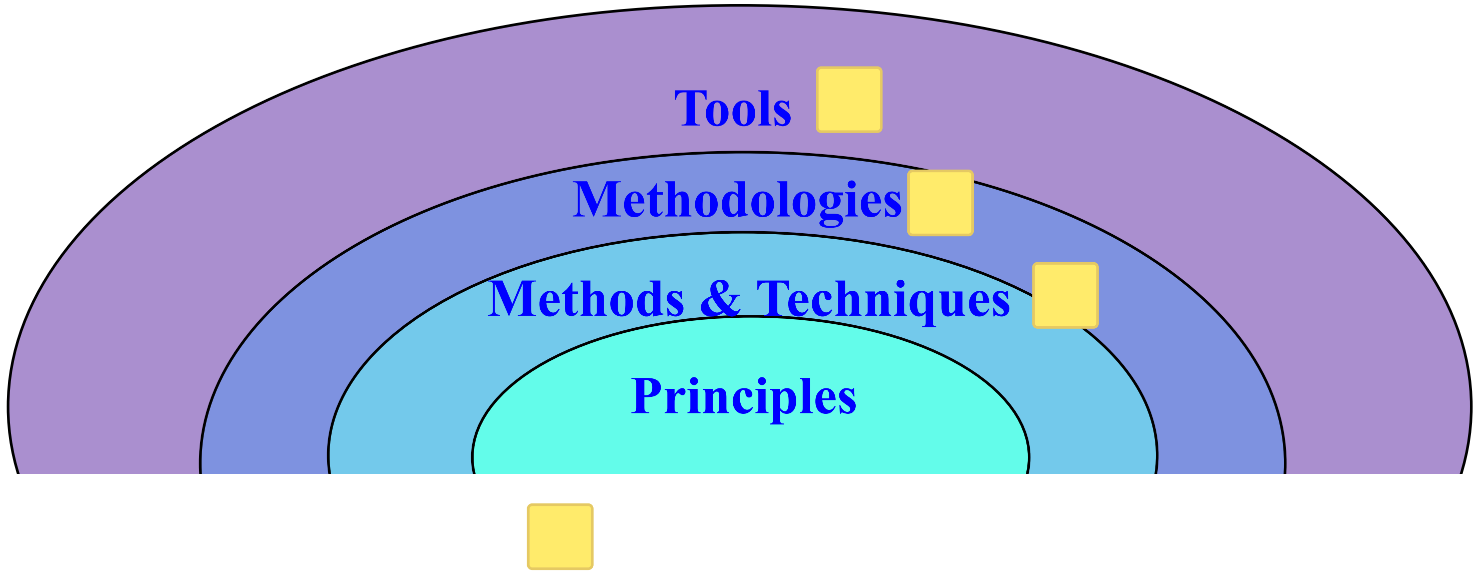
Fall 2018

Doo-Hwan Bae

SoC, KAIST

bae@se.kaist.ac.kr

Software Engineering Principles



Relationship between principles, techniques, methodologies and tool

What is SE Principle?

- Principle, "a fundamental, primary, or general law or truth from which others are derived"

  (source: http://www.dictionary.com/browse/principle)
- SE principle, "principles to be used in Software Engineering"
- Software Engineering, *"Application of* ***systematic, disciplined, and quantifiable approach*** *to the development, maintenance, and operation of software" (revisited)*

Why are SE Principles Important?

- Principles are relatively more stable than others
  - Principles < Methods, techniques < Tools
- Principles are not easily changed:
  - Once you master (understand) it, it is easy to follow the evolution of others.
  - Methods, techniques are researched & developed to satisfy principles better!
  - You may be able to predict future changes or evolutions!
- Example: What are three essential constructs (elements) common in modeling languages?

Software Engineering Principles

- Rigor and formality
- Separation of concerns
- Modularity
- Abstraction
- Anticipation of changes
- Generality
- Incrementality

Rigor and Formality

- Rigor: the quality or state of being very exact, careful, or strict
- Formality: the highest degree of rigor; based on formalism
- Rigor is a necessary complement to creativity.
  - Enhances creativity.

Q: What is the advantage of formality over rigor?

Q: Traditionally, there is only one phase of s/w development.
Where has a formal approach been used?

Separation of Concerns(1/3)

- Concept: Features overlap as little as possible
- Knitting a sweater, then threads are entangled
- How to untie threads?

Separation of Concerns(2/3)

- Software Engineering, *"Application of **systematic, disciplined, and quantifiable approach** to the development, maintenance, and operation of software"*
- Engineering approach
  - *Systematic*
  - *Quantifiable*
  - *Disciplined*
- Let us discuss, application of engineering approach to the 'untieing thread ' problem.

Separation of Concerns(3/3)

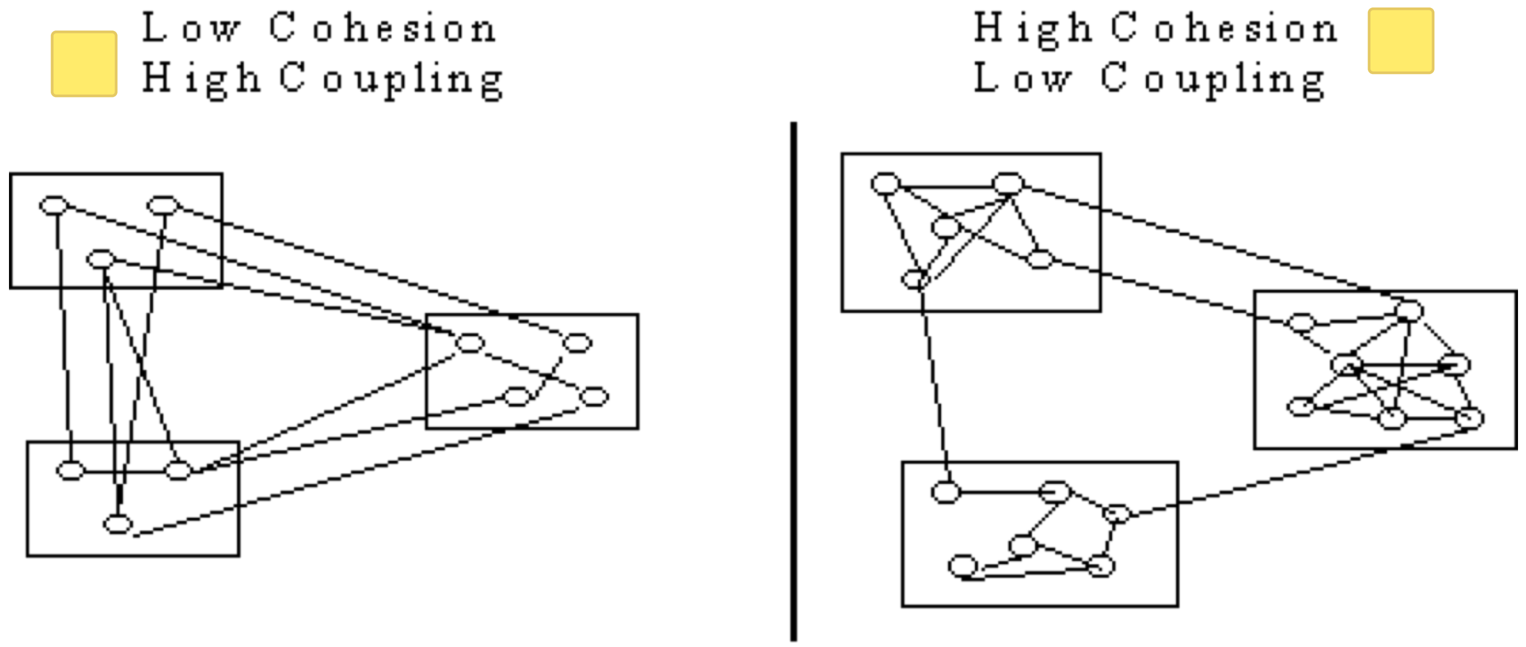
- A way to deal with inherent complexity
- Various ways in which concerns may be separated
  - In time
    Example :
  - In terms of qualities
    Example :
  - In different views of the software
    Example :
  - In terms of size
    Example :
  - In terms of responsibility
    Example :
- Q: May we miss some global optimization due to the separation of concerns?

Modularity(1/2))

- Divided into simpler pieces, called modules
- Top-down and bottom-up
- Decomposition
  - Divide and conquer
- Composition
    - Starting bottom up from elementary components
- To achieve modular design
  - High cohesion and low coupling
  - Low coupling: The degree of relatedness to other modules
  - High cohesion: The degree of the relatedness among internal elements of the module.

Modularity (2/2)

- Why is high cohesion, low coupling better?

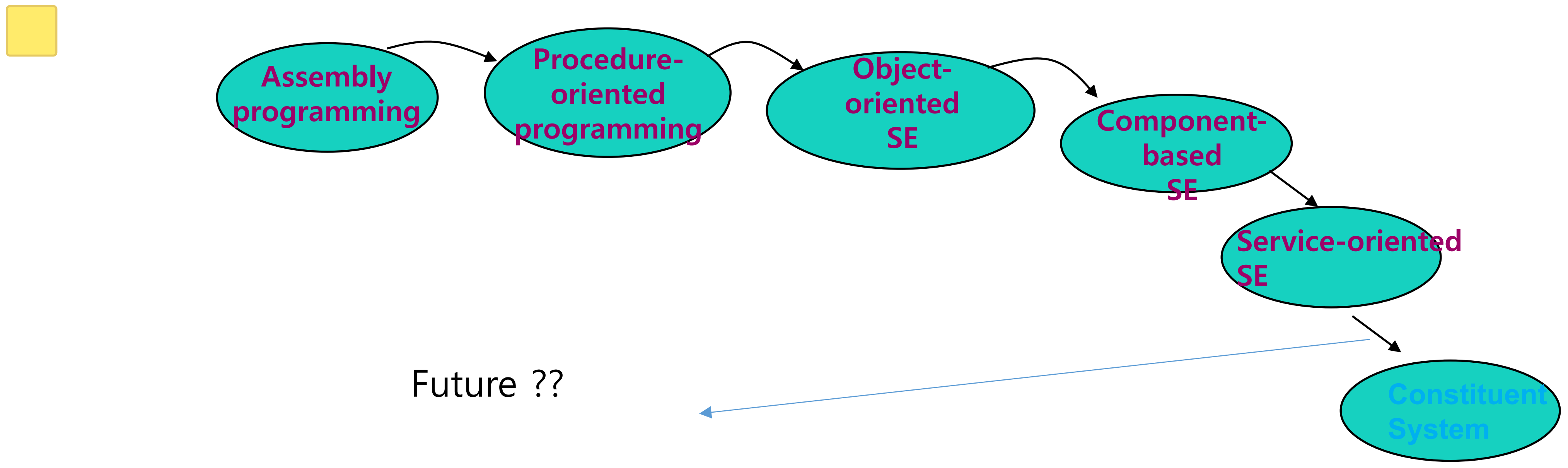


Abstraction(1/2)

- A process whereby we identify the important aspects of a phenomenon and ignore its details.
- Special case of separation of concerns
- History
  - Macro functions
  - Functions and procedures : called 'process abstraction'
  - Loops, selection: called 'control abstraction'
  - Abstract data types; called 'data abstraction'
  - Objects:
  - Components:
  - Design patterns: a set of classes with a certain behavior, commonly found in applications.
  - Services: UDDI, WSDL, SOAP, BPML, …
  - Systems:
- Helps us concentrate on the problem itself rather than the way on how to solve it.

Abstraction(2/2)

- SE paradigm evolution
  - Procedure -> Object -> Component -> Service -> ....



Anticipation of Change

- Software undergoes changes constantly.
- Likely changes should be isolated in specific portions.
- Affects maintainability, evolvability, reusability.

Generality

- Have more potential to be reused.
- Already provided by some off-the-shelf packages.
- Not necessarily more complex to solve a generalized problem than a specialized one
- More costly in terms of
  - Speed of execution
  - Memory requirements
  - Development time

Incrementality

- A process that proceeds in a stepwise fashion.
- Evolutionary process
- Example: Incremental system testing (thread testing)

Summary

- Conflicts among principles may exist
- Different methodologies emphasize different principles
- Your research contribution in SE principles is welcomed
  - Still, a lot of research/work need to be done
- Tools, methodologies, method/techniques will evolve more frequently than the principles
  - The principles may remain the same, I hope!!
- It is very IMPORTANT to understand SE principles because most of SE techniques/methods/tools are evolving accordingly (to satisfy them more!).