TATA CONSULTANCY SERVICES
Experience certainty.

TATA

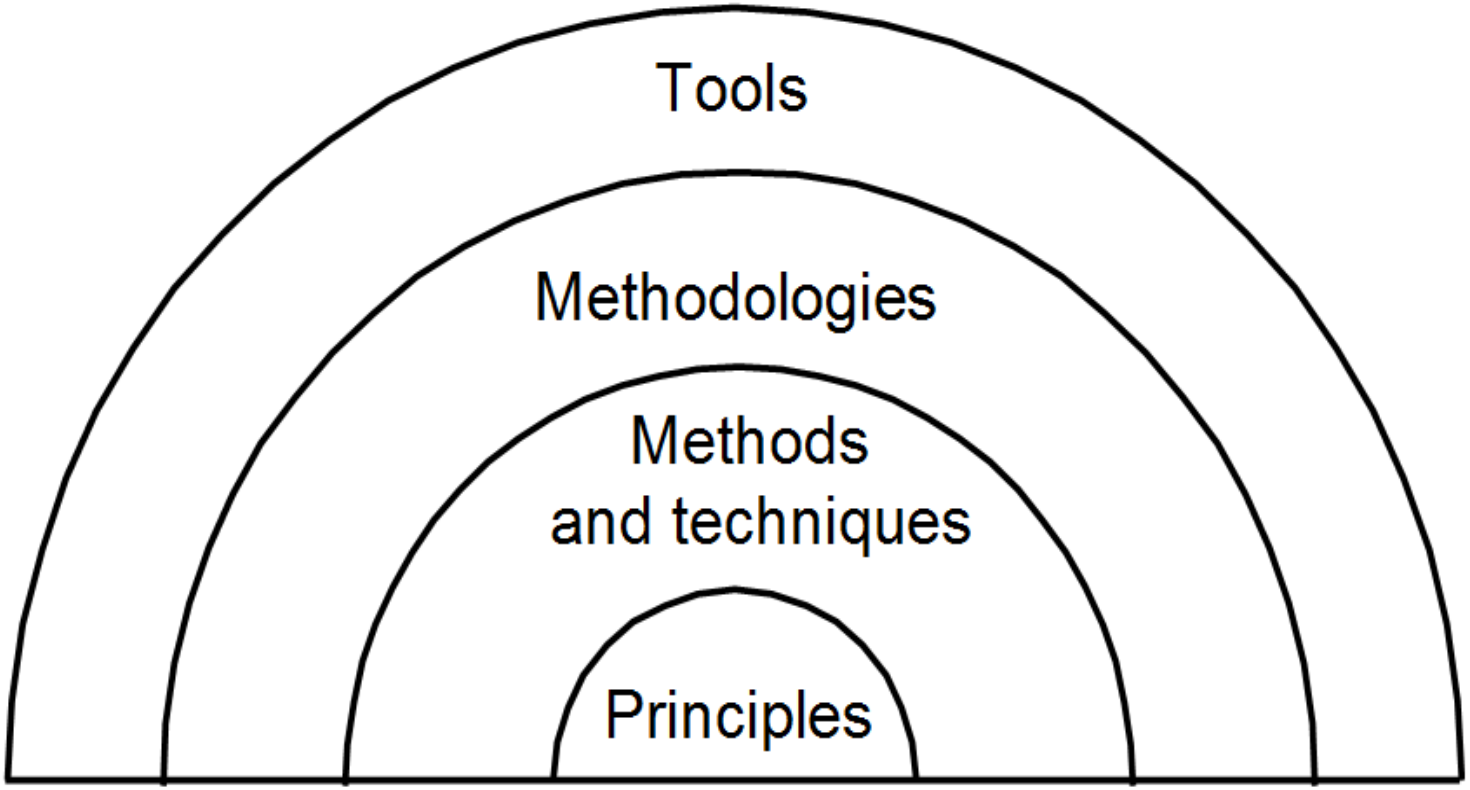
Software Engineering Concepts

Software Engineering Principles

- How do you achieve excellence in creating software products????

Software Engineering approach

A visual representation



What is Software Engineering?

- The IEEE Computer Society defines software engineering as:
  - "(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1)."

What is Software Engineering?

- Software engineering is about
  - the study of software process, development principles, techniques and notations
  - the production of quality software, that is delivered on time, within budget, and adequately meets its users' needs and expectations
  - the disciplined application of engineering, scientific and mathematical principles and methods in the economical production of quality software

Why is software engineering knowledge required?

- To predict time, effort, and cost
- To improve software quality
- To improve maintainability
- To meet increasing demands
- To lower software costs
- To successfully build large, complex software systems
- To facilitate group effort in developing software

Factors affecting software quality

- Complexity
  - No single programmer can understand it
  - Fixing one bug causes another one
- Change
  - Each change increases complexity and erodes the structure of a system
  - At some point, it is too expensive to make a change, and system cannot perform its function

How do you overcome the problems???

By adopting the software engineering principles….

Seven SE Principles

- Rigor and Formality
- Separation of Concerns
- Modularity
- Abstraction
- Anticipation of Change
- Generality
- Incrementality

Rigor and Formality

Examples: Product

- Mathematical (formal) analysis of program correctness
- Systematic (rigorous) test data derivation

Example: Process

- Rigorous documentation of development steps and
- assessment of timeliness

Separation of Concerns

- Allows one to deal with different aspects of a problem and concentrate on each aspect

separately

- Try to isolate issues that are less intimately related to the others
- When considering an issue separately, all details of related issues should not be considered

Separation of Concerns

Example: Process

- Go through phases one after the other (as in waterfall)
- Does separation of concerns by separating activities with respect to time

Example: Product

- Keep product requirements separate
  - functionality
  - performance
  - user interface and usability

Modularity

- Strategy to organize complex processes.
- How?
  - visible design rules
    - Architecture, Interfaces, Standards
  - hidden design parameters
- Example


15D 15D
15C

Benefits of Modularity

- The capability of decomposing a complex system into simpler pieces
- The capability of composing a complex system from existing modules
- The capability of understanding a system in terms of its pieces
- The capability of modifying a system by modifying only a small number of its pieces

Cohesion and Coupling

- Each module should be *highly cohesive*
  – module understandable as a meaningful unit
  – components of a module are closely related to one another
- Modules should exhibit *low coupling*
  – modules have low interactions with others
  – understandable separately

How to decompose

Step 1: Identify components

A good decomposition minimizes dependencies between components

| coupling - a measure of inter-component connectivity

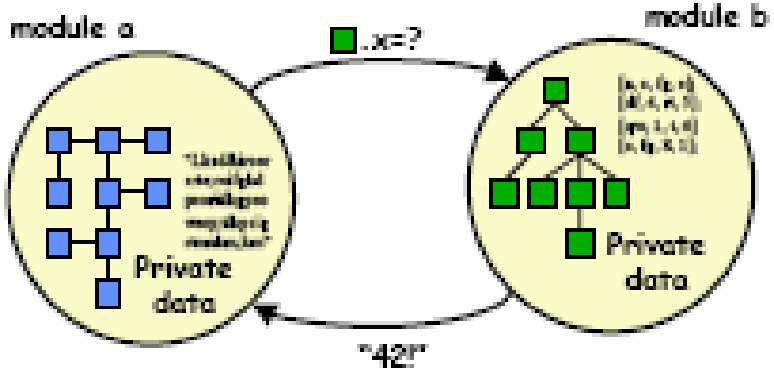
cohesion - a measure of how well the contents of a component go together

information hiding

having modules keep their data private

provide limited access procedures

this reduces coupling



Abstraction

- Identify the important aspects of a phenomenon and ignore its details
- Is a special case of separation of concerns
- The type of abstraction to apply depends on purpose
  - Example : the user interface of a watch (its buttons) abstracts from the watch's internals for the purpose of setting time;
  - Other abstractions are also needed to support repair


James Wendell George Pkwy
Spring St NW
Alexander St NW
Jones Ave NW
Ivy Peachtree Pl NW
Ralph McGill Blvd NE
Hardy Ivy Park
Centennial Olympic Park
Highland Ave NE
Freedom Pkwy
Andrew Young Int'l Blvd NE
Ellis St NE
Irwin St NE
Georgia World Congress Center
Georgia Dome
Philips Arena
Centennial Olympic Park Dr NW
Atlanta Journal Constitution Bldg
Five Points
Hurt Park
Edgewood Ave SE
Auburn Ave NE
Martin Luther King Jr NHS
Peachtree St NE
Courtland St NE
Boulevard NE
I-75 HOV Ln

Anticipation of Change

- For software to evolve gracefully it is necessary to anticipate how and where the changes are likely to occur
- Reusability is also strongly affected by anticipation of change
- Anticipation of change requires the appropriate tools to be available to manage the various versions and revisions of the software in a controlled manner.
- This is called configuration management

Generality

- While solving a problem, try to discover if it is an instance of a more general problem whose solution can be reused in other cases
- Carefully balance generality against performance and cost
- Sometimes a general problem is easier to solve than a special case

Incremental approach to solving problems

- Process proceeds in a stepwise fashion (*increments*)

Examples (process)

- deliver subsets of a system early to get early feedback from expected users, then add new features incrementally
- deal first with functionality, then turn to performance
- deliver a first prototype and then incrementally add effort to turn prototype into product

Activity

- Prepare a presentation using TCS standard template on The software engineering principles highlighting how they are applied during software development.
- You can explain the software engineering principles using your college project as an example.
- Submit your presentation on Decosystems.

How do we ensure software engineering principles are implemented?

- Principles become practice through methods and techniques
- methods and techniques are packaged in a *methodology*
- methodologies can be enforced by *tools*

Next Steps….

- Software Development processes and
- Software Development lifecycle models