BCA - First Semester
ubject Name- “Software Engineering”
Paper-BCA-103

By- Prof. Dileep Kumar Sahu

Assistant Professor

Department of Computer Application

Govt. Vishwanath Yadav Tamaskar Post Graduate Autonomous College, Durg (C.G.)

Email ID: dileepksahu20@gmail.com

Unit IV- Software Design & Coding

By- Prof DIleep Kumar Sahu, Assistant Professor, Govt. V.Y.T. PG
Auto. College Durg (C.G.)

Content

Principle of Software Design

- Partitioning
- Abstraction
- Top Down and Bottom up Strategies

nciple of Software Design

ware design principles are concerned with providing means to dle the complexity of the design process effectively.

ctively managing the complexity will not only reduce the effort led for design but can also reduce the scope of introducing errors ng design.

eloping design is a cumbersome process as most expansive errors often introduced in this phase.

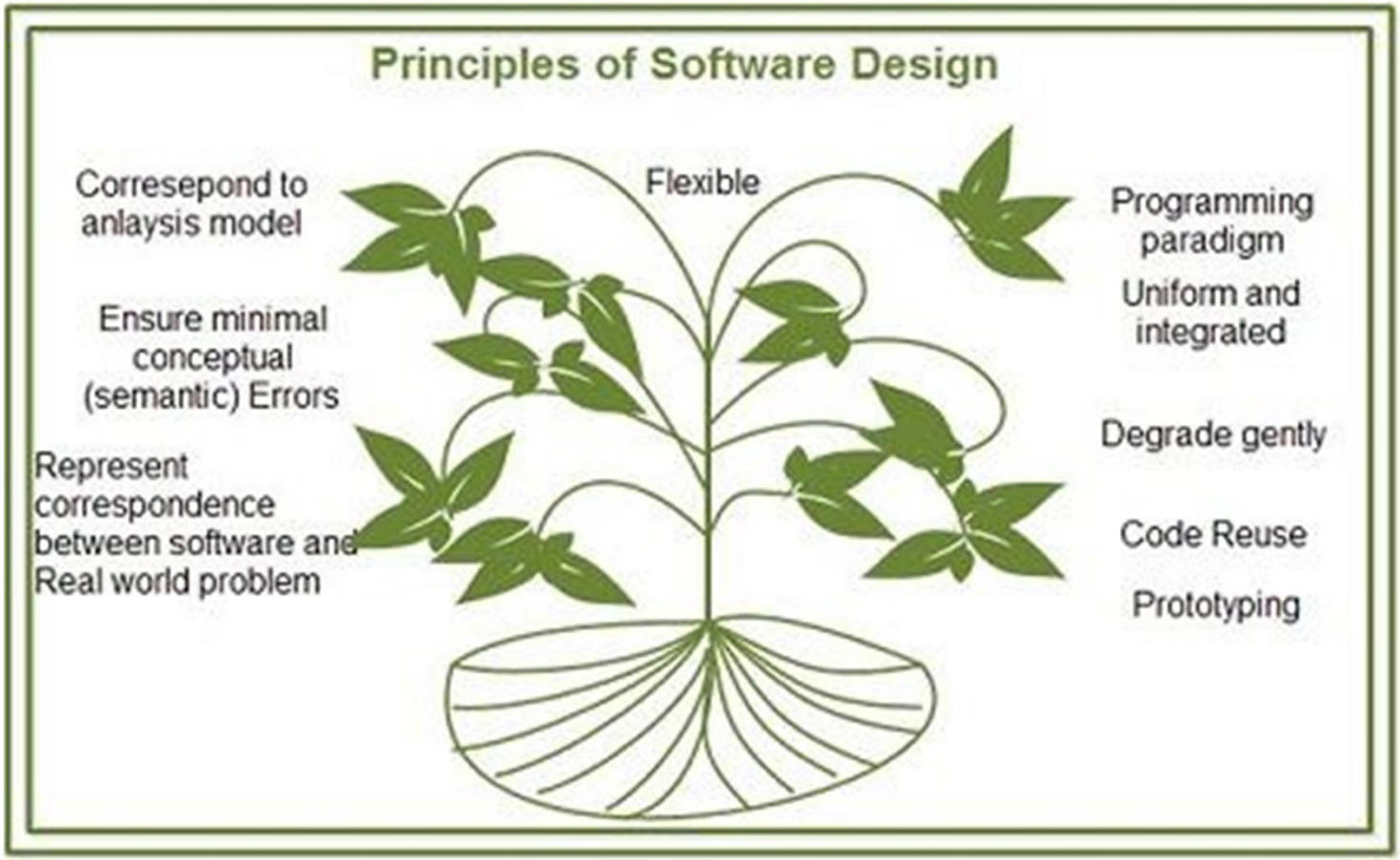
eover, if these errors get unnoticed till later phases, it becomes e difficult to correct them.

efore, a number of principles are followed while designing the ware.

e principles act as a framework for the designers to follow a good gn practice.

nciple of Software Design

COMMONLY FOLLOWED DESIGN PRINCIPLES



nciple of Software Design

ftware design should correspond to the analysis model: A design ment corresponds to many requirements, so, we must know how e design model satisfies all the requirements represented by the alysis model.

ect the right programming paradigm: The paradigm should be osen keeping constraints in mind such as time, availability of ources and nature of user's requirements.

ftware design should be uniform and integrated: For this, rules, mat, and styles are established before the design team starts signing the software.

nciple of Software Design

ftware design should be flexible: Software design should be xible enough to adapt changes easily.

To achieve the flexibility, the basic design concepts such as abstraction, refinement, and modularity should be applied effectively.

ftware design should ensure minimal conceptual (semantic) ors: such as ambiguousness and inconsistency are addressed in vance before dealing with the syntactical errors present in the sign model.

ftware design should be structured to degrade gently: Software ould be designed to handle unusual changes and circumstances, d if the need arises for termination, it must do so in a proper nner so that functionality of the software is not affected.

nciple of Software Design

ftware design should represent correspondence between the ftware and real-world problem: In such away that it always relates h the real-world problem.

ftware reuse: Software engineers believe on the phrase: 'do not nvent the wheel'. So, software components should be designed in ch a way that they can be effectively reused to increase the oductivity.

signing for testability: A common practice that has been followed to keep the testing phase separate from the design and plementation phases.

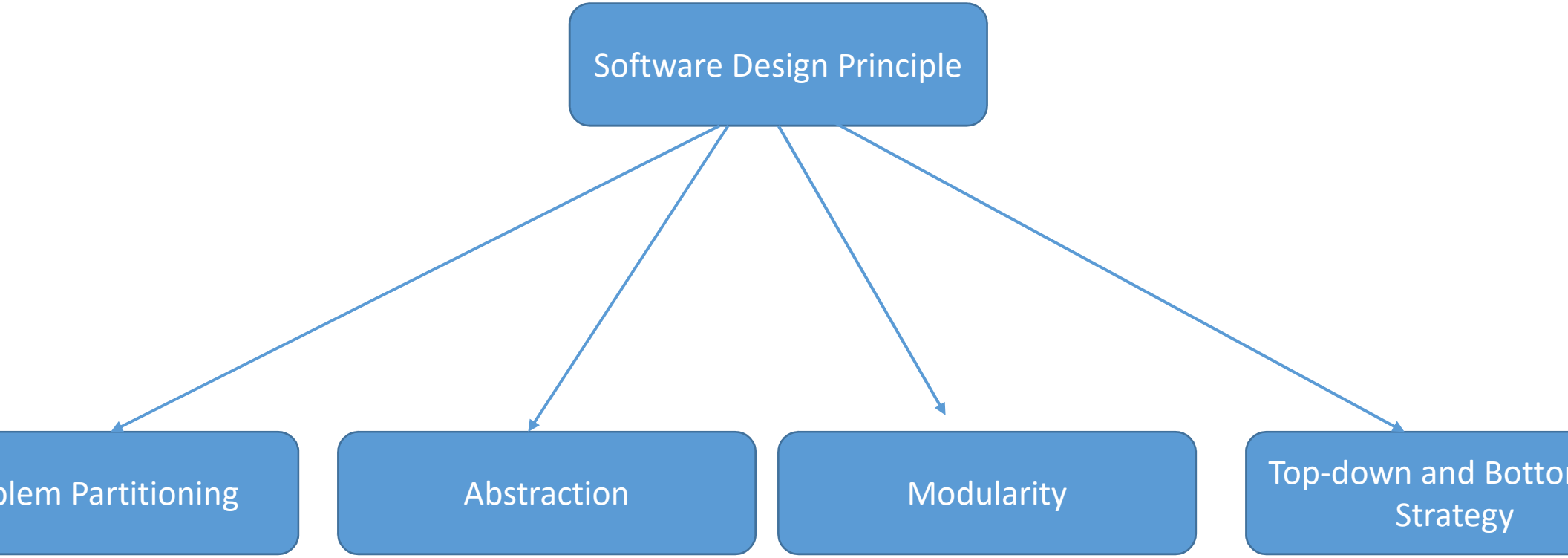
That is, first the software is developed (designed and implemented) and then handed over to the testers who subsequently determine whether the software is fit for distribution and subsequent use by the customer.

nciple of Software Design

ototyping: Prototyping should be used when the requirements are t completely defined in the beginning.

- The user interacts with the developer to expand and refine the requirements as the development proceeds.
- Using prototyping, a quick 'mock-up' of the system can be developed.
- This mock-up can be used as a effective means to give the users a feel of what the system will look like and demonstrate functions that will be included in the developed system.
- Prototyping also helps in reducing risks of designing software that is not in accordance with the customer's requirements.

nciple of Software Design



blem Partitioning

small problem, we can handle the entire problem at once but for the ficant problem, divide the problems and conquer the problem.

eans to divide the problem into smaller pieces so that each piece can aptured separately.

software design, the goal is to divide the problem into manageable es.

ts of Problem Partitioning:

ware is easy to understand

ware becomes simple

ware is easy to test

ware is easy to modify

ware is easy to maintain

ware is easy to expand

blem Partitioning

se pieces cannot be entirely independent of each other as they ether form the system.

y have to cooperate and communicate to solve the problem. This munication adds complexity.

e: As the number of partition increases = Cost of partition and plexity increases

traction

abstraction is a tool that enables a designer to consider a ponent at an abstract level without bothering about the internal ils of the implementation.

traction can be used for existing element as well as the ponent being designed.

e, there are two common abstraction mechanisms

- Functional Abstraction
- Data Abstraction

traction

ional Abstraction:

odule is specified by the method it performs.

details of the algorithm to accomplish the functions are not le to the user of the function.

ctional abstraction forms the basis for **Function oriented design roaches**.

Abstraction:

ails of the data elements are not visible to the users of data. Data traction forms the basis for **Object Oriented design approaches**.

dularity

ularity specifies to the division of software into separate modules
h are differently named and addressed and are integrated later on in
btain the completely functional software.

the only property that allows a program to be intellectually
ageable.

e large programs are difficult to understand and read due to a large
ber of reference variables, control paths, global variables, etc.

roperties of a modular system are:

module is a well-defined system that can be used with other
ications.

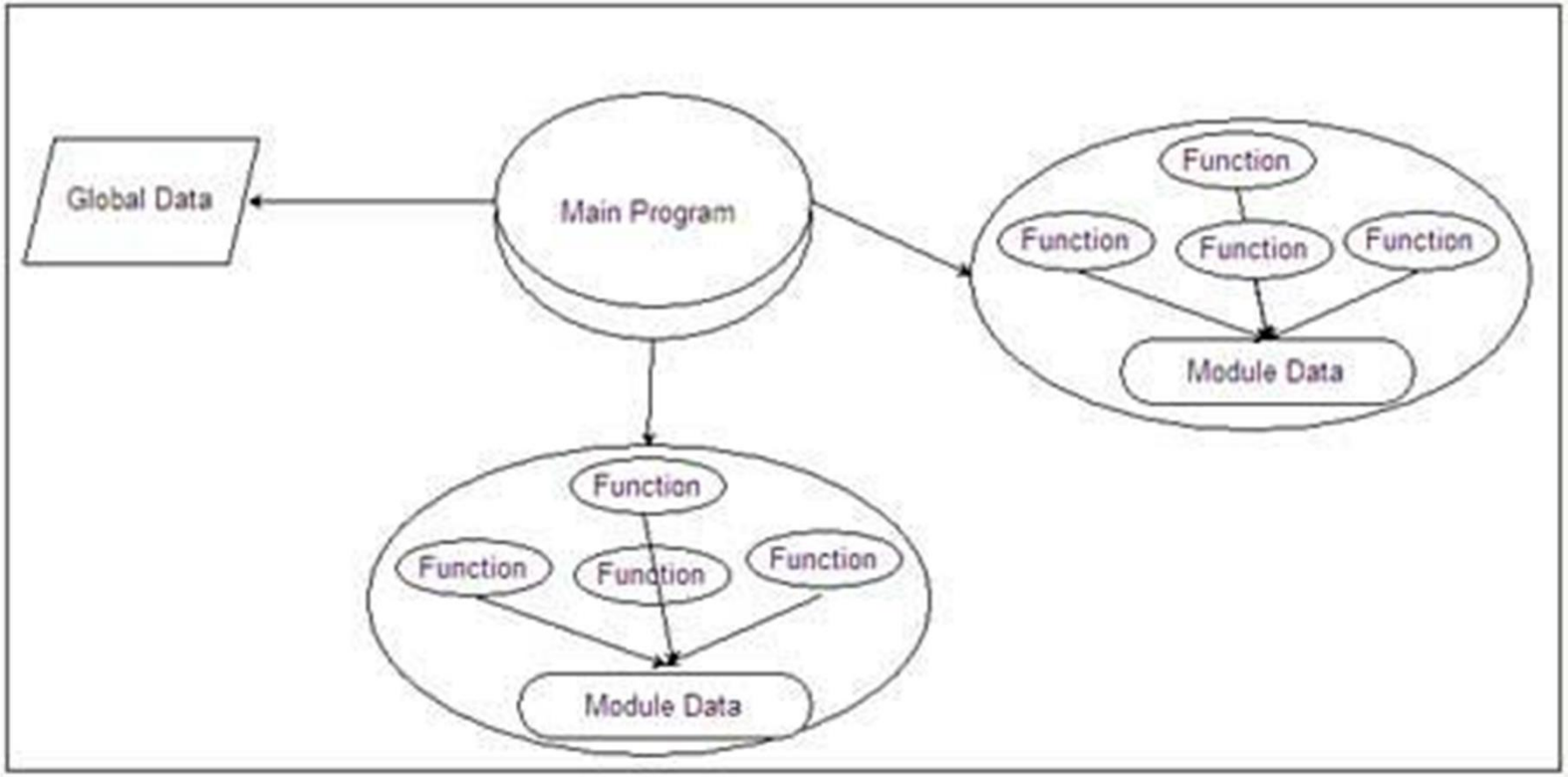
module has single specified objectives.

ules can be separately compiled and saved in the library.

ules should be easier to use than to build.

ules are simpler from outside than inside.

dularity


Global Data
Main Program
Function
Function
Function
Function
Module Data
Function
Function
Function
Function
Module Data

dularity

ntages of Modularity:

- ows large programs to be written by several or different people
- ncourages the creation of commonly used routines to be placed in library and used by other programs.
- mplifies the overlay procedure of loading a large program into n storage.
- ovides more checkpoints to measure progress.
- ovides a framework for complete testing, more accessible to test
- oduced the well designed and more readable program.

dularity

vantages of Modularity:

cution time maybe, but not certainly, longer

age size perhaps, but is not certainly, increased

pilation and loading time may be longer

r-module communication problems may be increased

e linkage required, run-time may be longer, more source lines
t be written, and more documentation has to be done

dularity: Modular Design

ar design reduces the design complexity and results in easier and faster nentation by allowing parallel development of various parts of a system. are two way to implements Modular design:

nctional Independence

ormation Hiding

ctional Independence: Functional independence is achieved by developing tions that perform only one kind of task and do not excessively interact with r modules.

ndependence is important because it makes implementation more accessible and faster.

he independent modules are easier to maintain, test, and reduce error propagation and can e reused in other programs as well.

hus, functional independence is a good design feature which ensures software quality.

easured using two criteria:

sion: It measures the relative function strength of a module.

ling: It measures the relative interdependence among modules.

dularity: Modular Design

ormation hiding: Modules should be specified that data include in a module is inaccessible to other modules that do not need for n information.

benefits of information hiding is that when modifications are uired during testing's and later during software maintenance.

s is because as most data and procedures are hidden from other s of the software, inadvertent errors introduced during lifications are less likely to propagate to different locations within software.

tegy of Design

ood system design strategy is to organize the program modules in
a method that are easy to develop and latter too, change.

ctured design methods help developers to deal with the size and
plexity of programs.

esign a system, there are two possible approaches:

p-down Approach

ottom-up Approach

-down Approach

n system is divided into several subsystems and components. Each he subsystem is further divided into set of subsystems and ponents.

process of division facilitates in forming a system hierarchy cture.

complete software system is considered as a single entity and in tion to the characteristics, the system is split into sub-system and ponent.

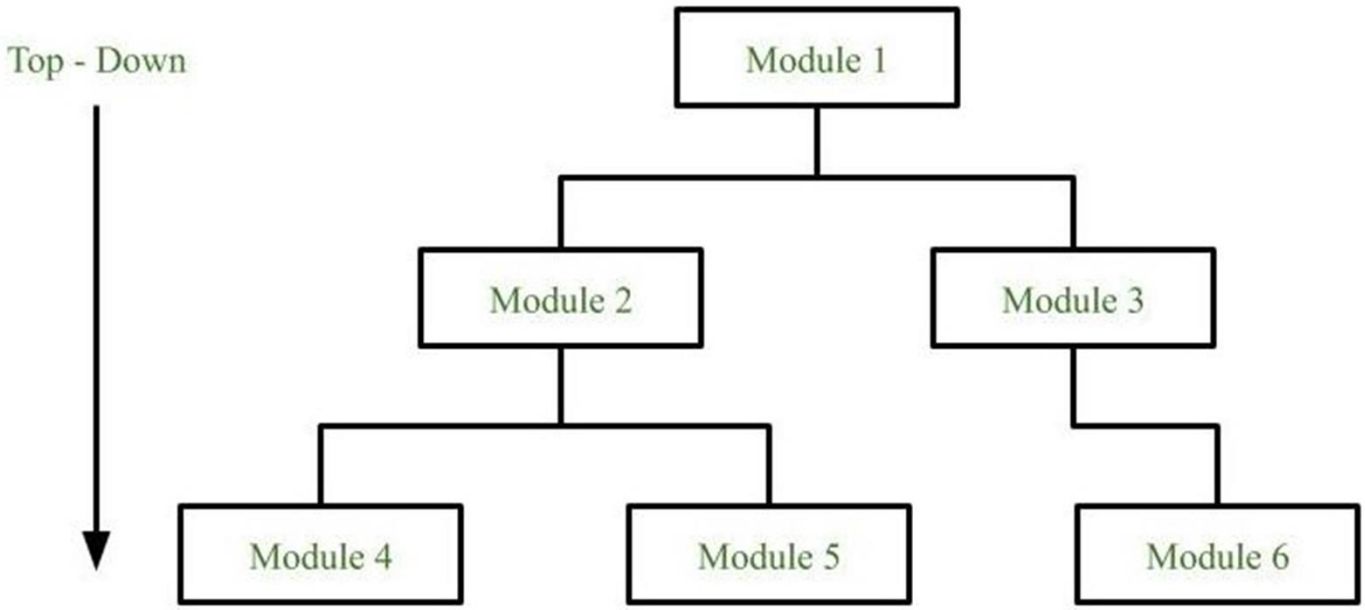
same is done with each of the sub-system.

process is continued until the lowest level of the system is hed.

-down Approach

design is started initially by defining the system as a whole and
n keeps on adding definitions of the subsystems and components.

en all the definitions are combined together, it turns out to be a
plete system.



-down Approach

ntages:

main advantage of top down approach is that its strong focus on uirements helps to make a design responsive according to its uirements.

vantages:

ect and system boundries tends to be application specification nted. Thus it is more likely that advantages of component reuse be missed.

system is likely to miss, the benefits of a well-structured, simple itecture.

tom-up Approach

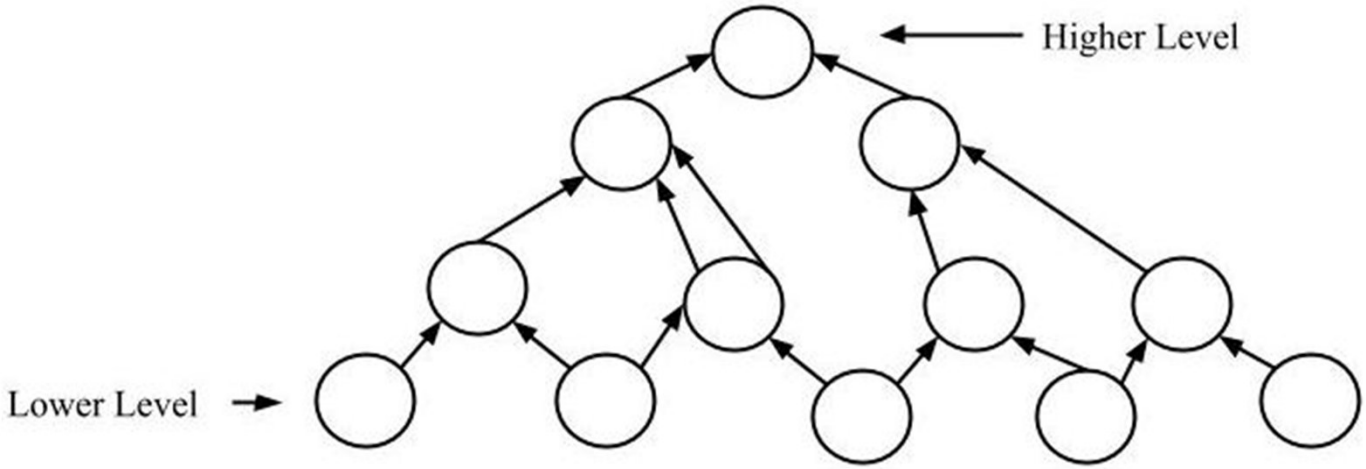
esign starts with the lowest level components and subsystems.

sing these components, the next immediate higher level onents and subsystems are created or composed.

process is continued till all the components and subsystems are osed into a single component, which is considered as the lete system..

ing the basic information existing system, when a new system to be created, the bottom up strategy suits the purpose.

tom-up Approach



tom-up Approach

ntages:

economics can result when general solutions can be reused.

n be used to hide the low-level details of implementation and be ged with top-down technique.

vantages:

not so closely related to the structure of the problem.

quality bottom-up solutions are very hard to construct.

erence

s://www.javatpoint.com/software-engineering-software-design-ciples

://ecomputernotes.com/software-engineering/principles-of-ware-design-and-concepts

Thank You

By- Prof DIleep Kumar Sahu, Assistant Professor, Govt. V.Y.T. PG
Auto. College Durg (C.G.)