DESIGNING PROBLEMS:

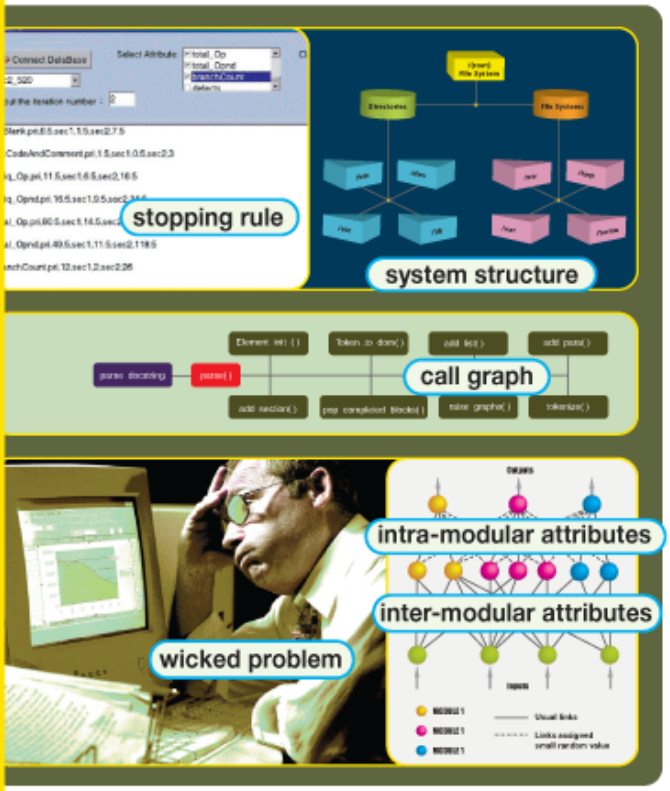
Date: 14 June

Subject: A Little Planning Leads to a Big Payoff in Software Design

Every engineer worries about **wicked problems.** Without **stopping rules**, engineers often don't know whether problems are fixed or not. But I'd like to point out that these problems can be easily avoided. Programmers just need to use **call graphs** to depict **system structures**. These let the programmer eliminate problems before they even begin.

Some programmers are geniuses with code, but they're unfamiliar with call graphs. This is unfortunate, because these systems greatly **simplify** the design process. Software planning provides programmers with an **abstraction** of the final product. Abstract systems are described in terms of their **modularity**, **cohesion**, and **coupling**. Programmers can even make allowances for **information hiding**.

When the plan is finished, the programmer can examine the **inter-modular attributes** and the **intra-modular attributes.** Errors can be eliminated while the software's **complexity** is low. Then, engineers are less likely to encounter complex problems later on.



wicked problem | information hiding | call graph | simplify | stopping rule

1 A(n) _____ can have multiple causes and may be difficult to solve.

2 A(n) _____ shows the basic structure of how a system will work.

3 Modules conceal information from each other in a process called _____.

4 A problem without a(n) _____ may be difficult or impossible to solve.

5 Use of systems and procedures can _____ complicated processes.

1 cohesion / coupling

A) _____ describes the strength of connections between modules.

B) _____ is the connection between modules in a system.

2 inter-modular attributes / intra-modular attributes

A) Characteristics of individual modules are _____.

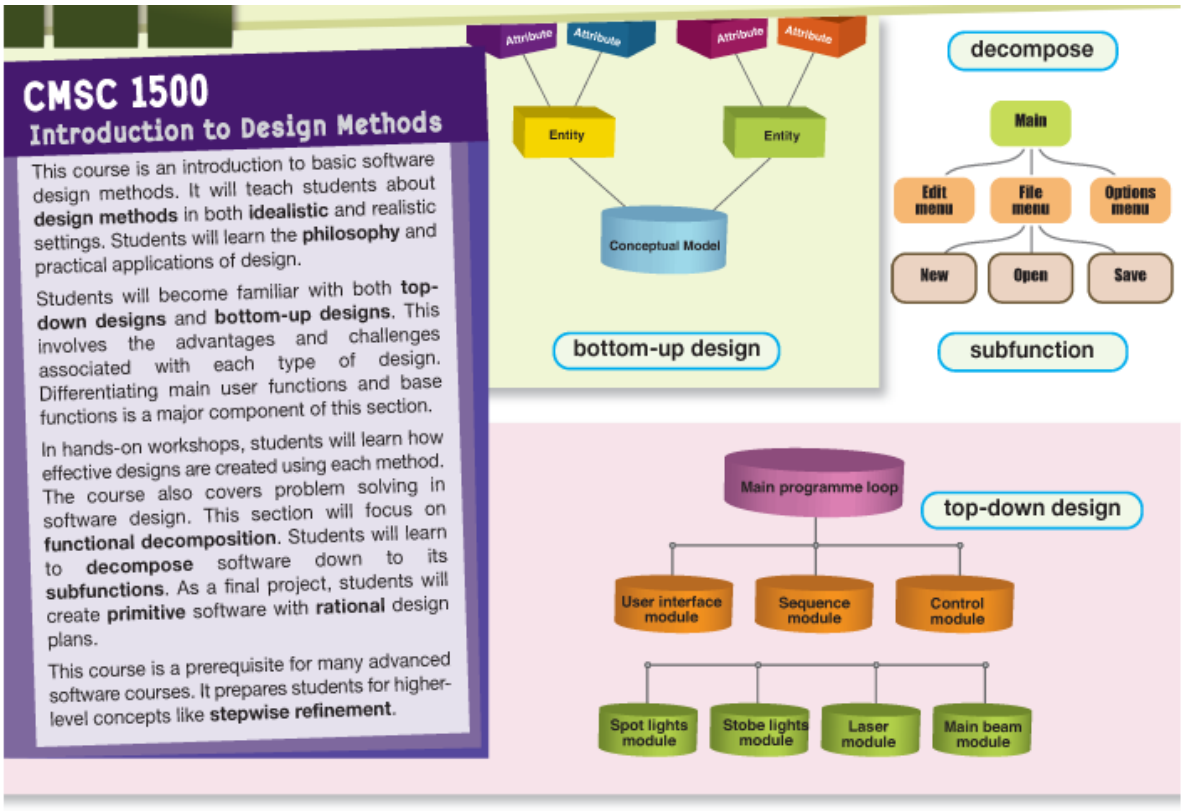
B) _____ describe the characteristics of an entire system.

3 system structure / abstraction

A) A(n) _____ is the network of connections between modules.

B) A(n) _____ ignores details.

DESIGNING METHODS



1 top-down design / bottom-up design

A) A _____ begins with the main user functions.

B) A _____ begins with the most primitive functions.

2 primitive / idealistic

A) A(n) _____ person ignores possible problems that may arise.

B) If a software is _____, it is very simple.

3 philosophy / subfunction

A) A _____ is combined with others to make a function.

B) _____ provides a means of viewing the world.

3 Choose the sentence that uses the underlined part correctly.

1
- ○ **A)** Engineers use a design method when designing new software systems.
- ○ **B)** To decompose software is to make it more complicated.

2
- ○ **A)** Functional decomposition can be used to reveal and eliminate problems in software.
- ○ **B)** Stepwise refinement is a method of creating a plan for new types of software.

3
- ○ **A)** Engineers can use a process called stepwise refinement to solve problems in existing software.
- ○ **B)** In an ideal world, software designs will never follow a rational procedure.

4
- ○ **A)** A subfunction is a higher-order function.
- ○ **B)** When an engineer decomposes a function, he or she is left with smaller subfunctions.

5
- ○ **A)** It is important that engineers follow a rational procedure.
- ○ **B)** A philosophy is primarily concerned with the real-world application of ideas.