DATE
PAGE

[redacted]

Ques. 1. What does AOP address?

A:. AOP basically addresses the goals for a particular project that are easy to develop and maintain, and also it should make efficient use of memory. In addition to this, AOP can also be used to solve those programing problems that cannot be solved by both OOP and procedural language.

AOP helps in writing clear programs involving such aspects, including appropriate isolation, composition and reuse of aspect code.

Ques. 2. How is it done?

A:. To understand how AOP addressing is done, let us take the example of black- and white image processing systems, in which our aim is to design a desired domain model through a series of filters to produce some desired output system.

First step would be to use a set of primitive procedures that would implement basic filters, and higher level filters would be defined in terms of primitive ones. For example:

a primitive or filter, which takes two images and returns their pixelwise logical, ~~or might be implemented as~~ The programmer can start from or1 and other primitive filters. and can select those black pixels on a horizontal edge.

The second step would be to take a global perspective of a particular program, and map out intermediate results and finally coding up a version of program that fuses loops appropriately to implement the original functionality while creating as few intermediate images as possible.

The final step would be to implement cross-cutting phenomena, which is responsible for code tangling. The single composition mechanism the language provides us — procedure calling — is very well suited to building up the un-optimized functional units.

Ques.3. List the elements of AOP with a brief definition/description of each.

A:: The elements of AOP are —

1) A component language with which to program the components.
2) One or more aspect languages with which to program the aspects.
3) An aspect weaver for the combined languages.
4) A component program, that implements the components using the component language.
5) One or more aspect programs that implement the aspects using the aspect languages.

(A) THE COMPONENT LANGUAGE AND PROGRAM.

Designing an AOP system involves understanding what must go into the component language, what must go into the aspect languages, and what must be shared among languages. The component language must allow the programmer to write at the same time ensuring that those programs do not pre-empty anything the aspect programs need to control. The aspect languages must support implementations of desired aspects, in a natural and concise way.

(B) THE ASPECT LANGUAGE AND PROGRAM

Communication aspect programs would like to be able to control the amount of copying of arguments that takes place when there is a remote method invocation. To do this, the aspect language must effectively allow them to step into implementation of method invocation.

to detect whether it is local or remote, and to implement the appropriate amount of copying in each case.

(C) ASPECT WEAVER

Aspect weaver must process the component and aspect languages, composing them properly to give the result as desired total system operation. The concept of join points is also important to the function of aspect weaver. These elements of component language semantics that the aspect programs join with.

(✗)