Task 1: Terminology of Aspect-Orientation

a) aspect: extensions of the code that get weaved into the software, it can extend or substitute code.

b) pointcut: tells the program which joint points are used from the rest of the program, advice is added to the code snippet selected by the pointcut

c) joint point: part of the code where extension of the code is possible. Aspects can be woven in here.

d) advice: code snippet that is added at a joint point and chosen by a pointcut.

e) inter-type declaration: static extension mechanism, that can add member variables, interfaces or methods to classes.

f) quantification: by using wildcards in the pointcuts many joint points will get code from the aspect.

Task 4: Obliviousness Principle

a) Work division between aspect and base program programmers possible since the program can be implemented without thinking about the additional features. The aspects can later be implemented.

b) Low preplanning effort to the cost of reducing information hiding.

c) The fragile-pointcut-problem describes the issue of changing joint point names in the base program, the aspects therefore will not be able to access the joint points anymore because the point cuts do not point to something anymore.