**16)** What do you mean by *High level breaks*? We found some example but not a general definition, so we are not clear about it yet.

I just put a post on BeeP discussing this very point

**19)** We understand that the commented out code must have reasons to be omitted, but it is not clear to us the thing about the date.

commented out code should contain a date since if its too old it must be removed.

**22)**What do you mean with *External program interfaces*? Does it refer to application outside of Glassfish that the class is using? Or classes/components in other modules/packages?

your first guess, obviously :)

**27)**In this point, you talk about *long methods* and *big classes*. How long (big) is *long (big)?*Isn't it somehow subjective?

With respect to encapsulation, coupling and cohesion, well I know what they mean but they are also kind of subjective in some cases. Also I think that we would need a non trivial perspective of the Glassfish that goes beyond the class that was assigned to us. So maybe we are thinking of it in the wrong way, so I would like you to clarify this.

**Average of class and method sizes.**

both of these points are to be evaluated against the mean average size that you can observe for the rest of the system… so the question is, are there any very long classes with respect to the rest of the classes in the set?

**28)**How do we state whether the type or the visibility are correct?

I mean, beyond checking that a variable which is delcared as *int* is not used in operations only possible on *String*, for exmple, in what other ways is the correctness checked?

And for checking the visibility of the attributes in our class, we think we would have to check for the usages of such variables in other classes, since in the same class any visibility creates no problem. Is this what you are expecting?

precisely, this is indeed what we are expecting.

**Make Netbeans to check it.**

**29)** Which one is the *correct* scope?

Does it mean checking that variables are not used in a scope in which they do not exist?

at least this, yes

Or that they are declared in a scope which is too "global" i. e. they can be declared in a smaller scope without generating errors?

if this is done as well, even better

**35)**Is this point intended to find typing errors? Because we also understood it as putting ourselves in the place of the person who wrote the code and checking that the method was the right one to be called.

both points of view are applicable, it basically depends on the nature of the method

**49)**Like in the previous one, does it refer only to a syntactic revision? Or do we have to check that the appropriate comparisons and operators are being used, according to the functional objective?

same here. it depends on the nature of the method. if its result is wrong if the appropriate comparisons and operations do not work then additional review and care should be invested, possibly leading to refactoring