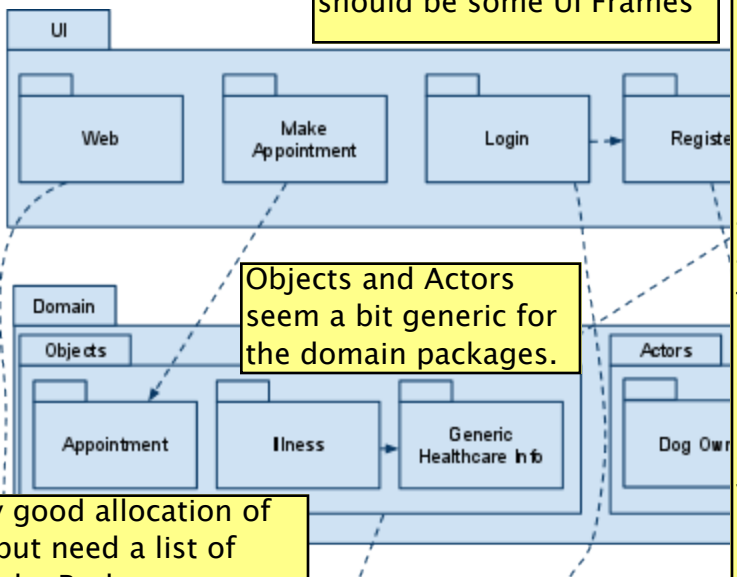
7.6/10 - Reasonably good job. You have a reasonable structural understanding of using UML through classes and packages, and even basic SDs and CDs. In some cases, the semantics of the relationships are not yet being conveyed using associations, dependencies, aggregations and the like.

In a Logical Design (aka high-level design), it is important to identify and organize the elements that at a coarse-level will be used to execute the capabilities outlined in the requirements. The starting point is the analysis model and you add the next level of organization - the assembly of things into packages and layers. Further, each package must be elaborated into a Design Class Diagram (DCD). Simultaneously, you need to be addressing the dynamic elements of the system through the refinement and elaboration of the interaction diagrams. Note that this has not started on physical design yet. That will impose some of its own requirement and constraints at a later state.

Note in the Domain layer, we are still being quite "logical or conceptual" in our descriptions. We take the analysis models like the DM and detail the classes into a DCD. At the same time, we identify new objects that will need to be modeled for the interactions from the SSDs. Further, we organize these into packages and have DCDs for those classes now combined into a package. Of course, details from the UCs, the OCs, and the more detailed SDs and CDs enable us to record attribute associations, dependencies, aggregations, and the like.

Please look at my more detailed comments for specifics and remember that they are meant to be constructive and helpful.

Login and the others should be some UI Frames



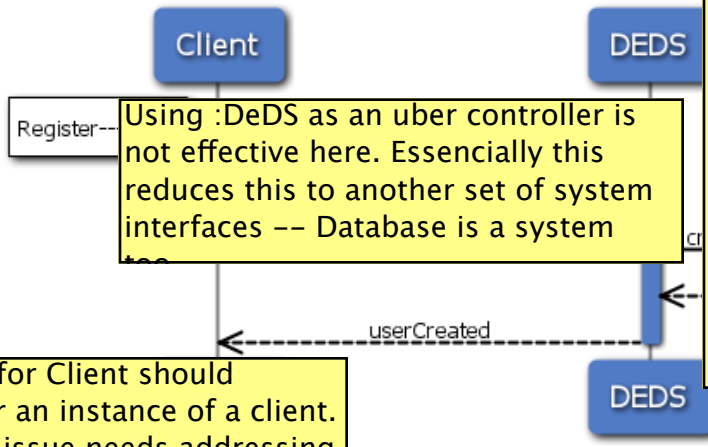
Objects and Actors seem a bit generic for the domain packages.

Reasonably good allocation of packages, but need a list of classes for the Packages.

Most of the Dependencies are understandable, but it is often a good job to provide a textual description or annotation to clarify

Like the models we have been doing in the class, Logical Architecture should have a textual description outlining your rationale and assumptions. This is often a good place to discuss dependencies too.

Dog Owner Registra



Using :DeDS as an uber controller is not effective here. Essencially this reduces this to another set of system interfaces -- Database is a system too.

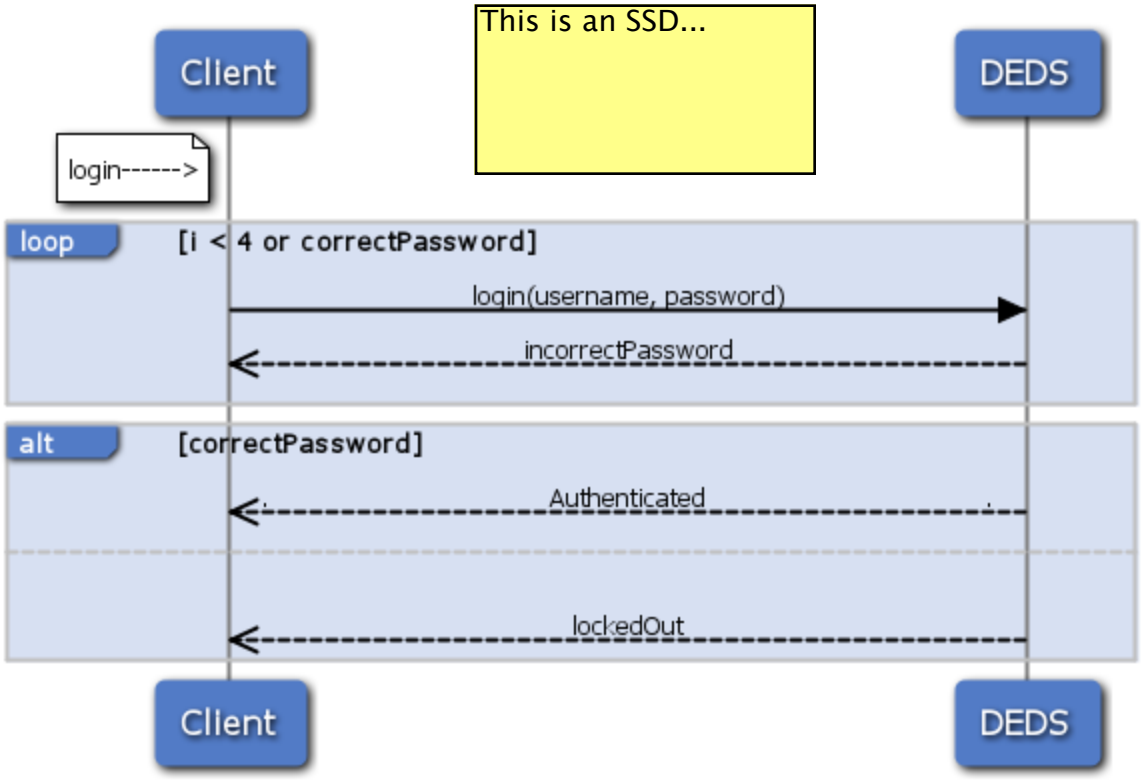
The syntax for Client should be :Client or an instance of a client. This syntax issue needs addressing throughout the document.

Database as a raw object is a bit premature design unless there is no other element that processes the message. I do not think that the database handles createUser... Perhaps :Registry or :Accounts would suffice.

note left of Client: Register--------->
Client->DEDS: register(name, address, email, dogs)
activate DEDS
DEDS->Database: creat ail, dog
Database-->DEDS: use
DEDS-->Client: userCre

I like the idea of the text, but a text version of the diagram is not what I had in mind

Login System

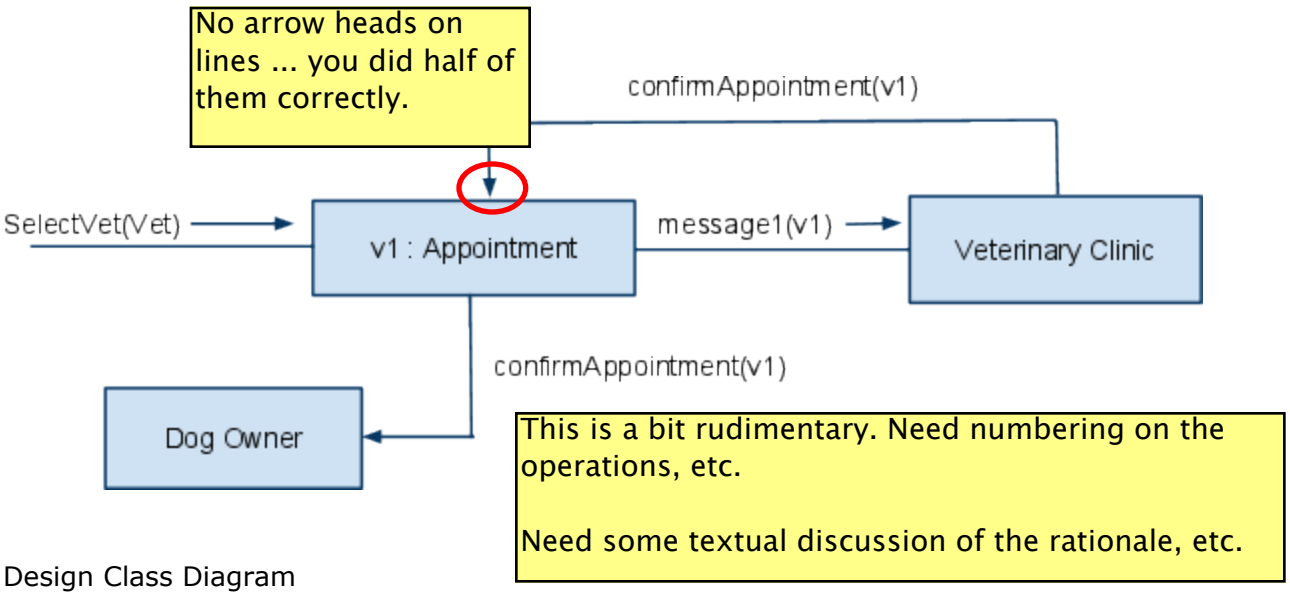


note left of Client: login------>

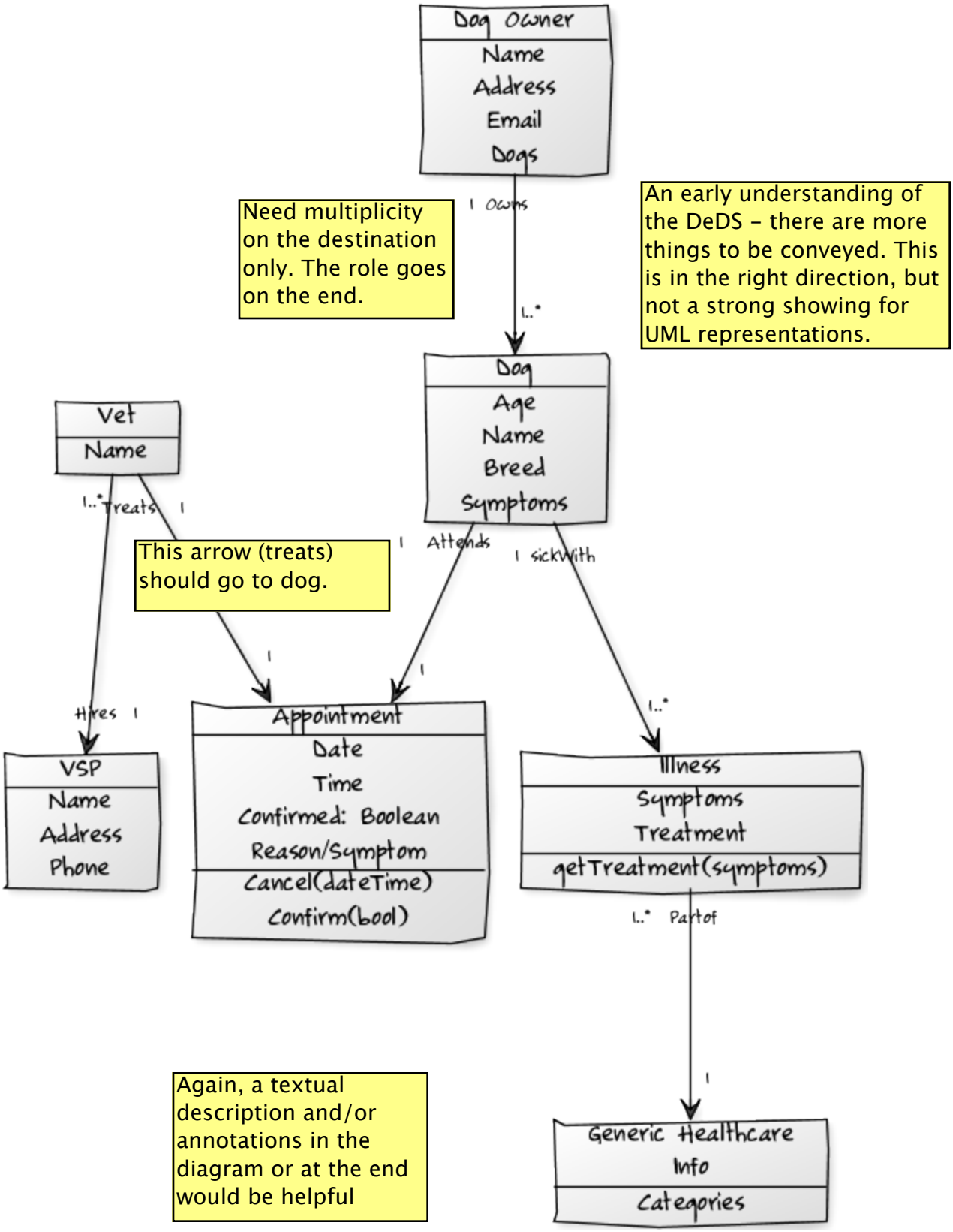
loop i < 4 or correctPassword
    Client->DEDS: login(username, password)
    DEDS-->Client: incorrectPassword
end
alt correctPassword
    DEDS-->Client: . Authenticated .
else
    DEDS-->Client: lockedOut
end

Again, I like the idea of the text, but a text version of the diagram is not what I had in mind

Appointment Scheduling:



Design Class Diagram


Dog Owner
Name
Address
Email
Dogs
1 Owns
1..*
Need multiplicity on the destination only. The role goes on the end.
An early understanding of the DeDS – there are more things to be conveyed. This is in the right direction, but not a strong showing for UML representations.
Dog
Age
Name
Breed
Symptoms
Vet
Name
1..* Treats 1
This arrow (treats) should go to dog.
1 Attends
1 sickWith
1
1
Hires 1
1..*
VSP
Name
Address
Phone
Appointment
Date
Time
Confirmed: Boolean
Reason/Symptom
Cancel(dateTime)
Confirm(bool)
Illness
Symptoms
Treatment
getTreatment(symptoms)
1..* Partof
1
Again, a textual description and/or annotations in the diagram or at the end would be helpful
Generic Healthcare Info
Categories