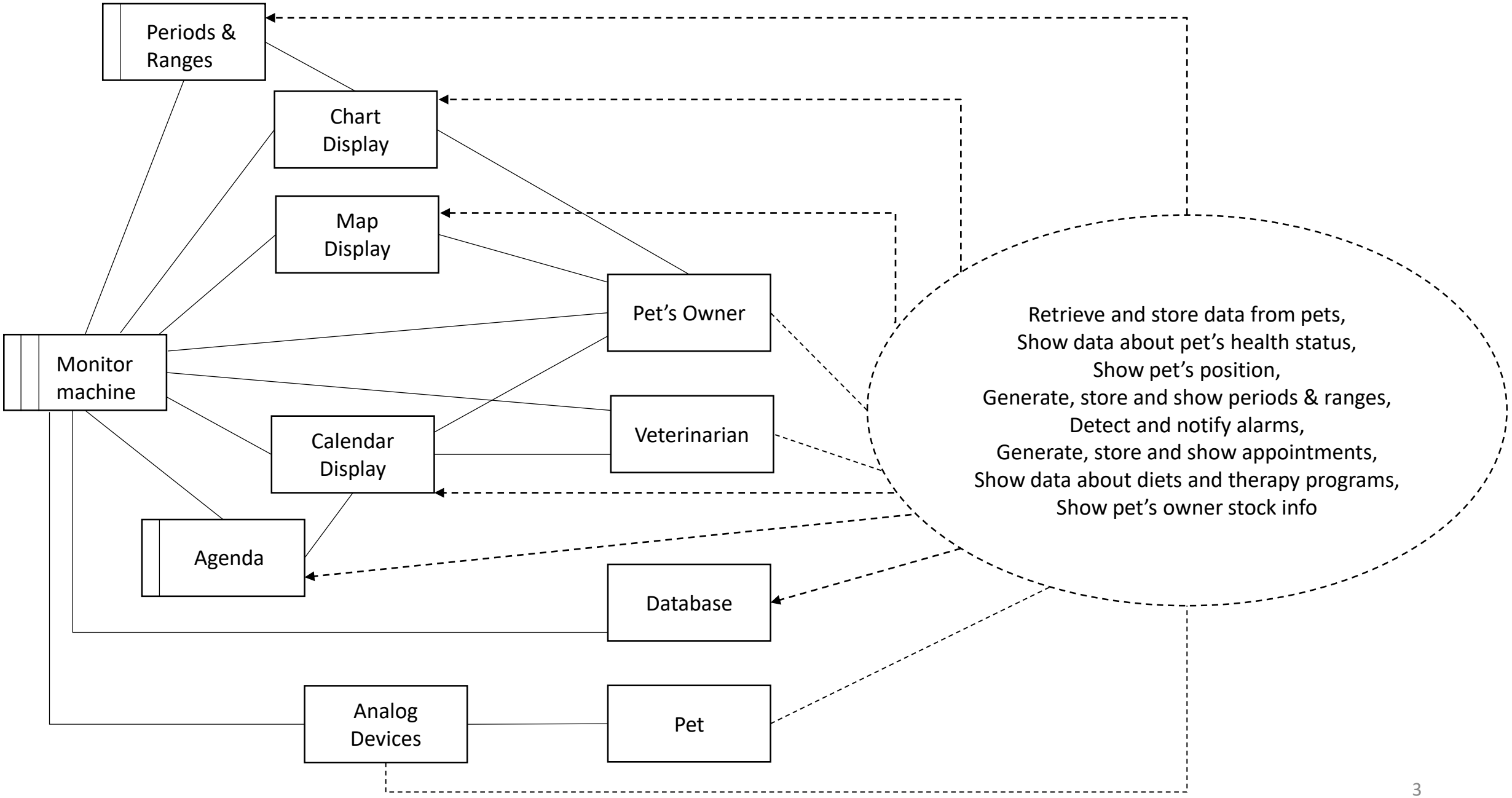
Requirements Engineering 2019/20

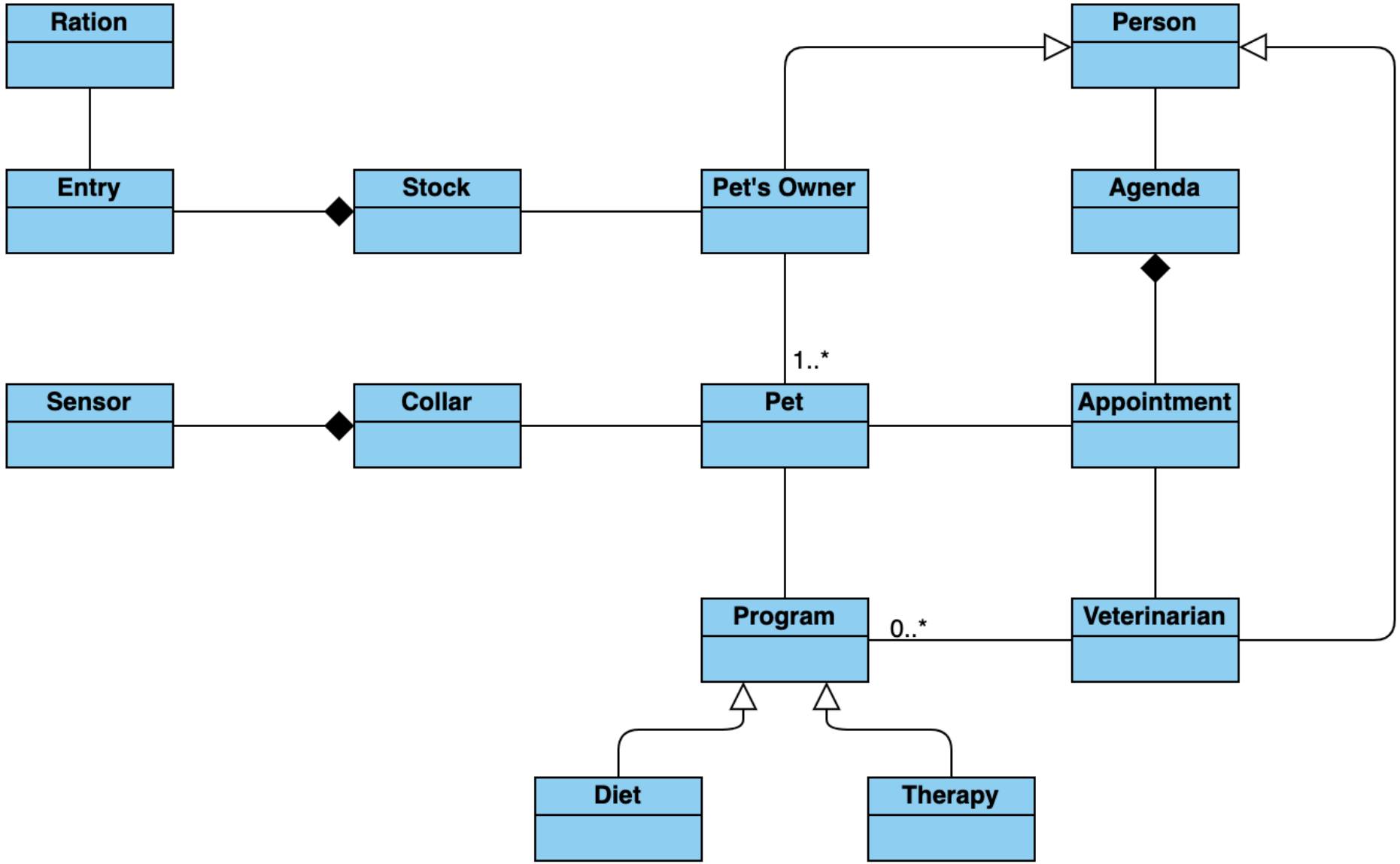
MyPet

Laino Lorenzo
Gatto Marco

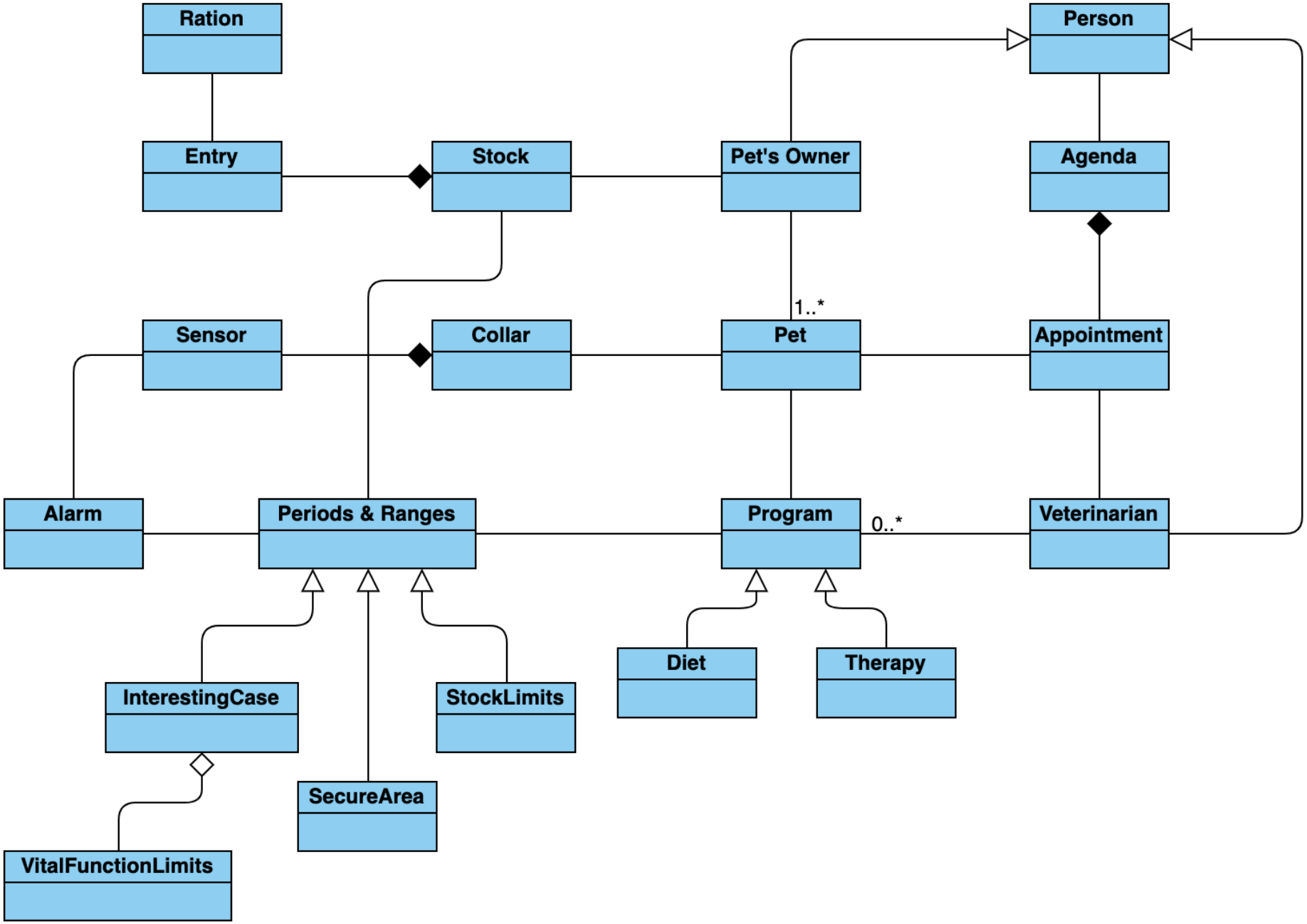
Problem Diagram

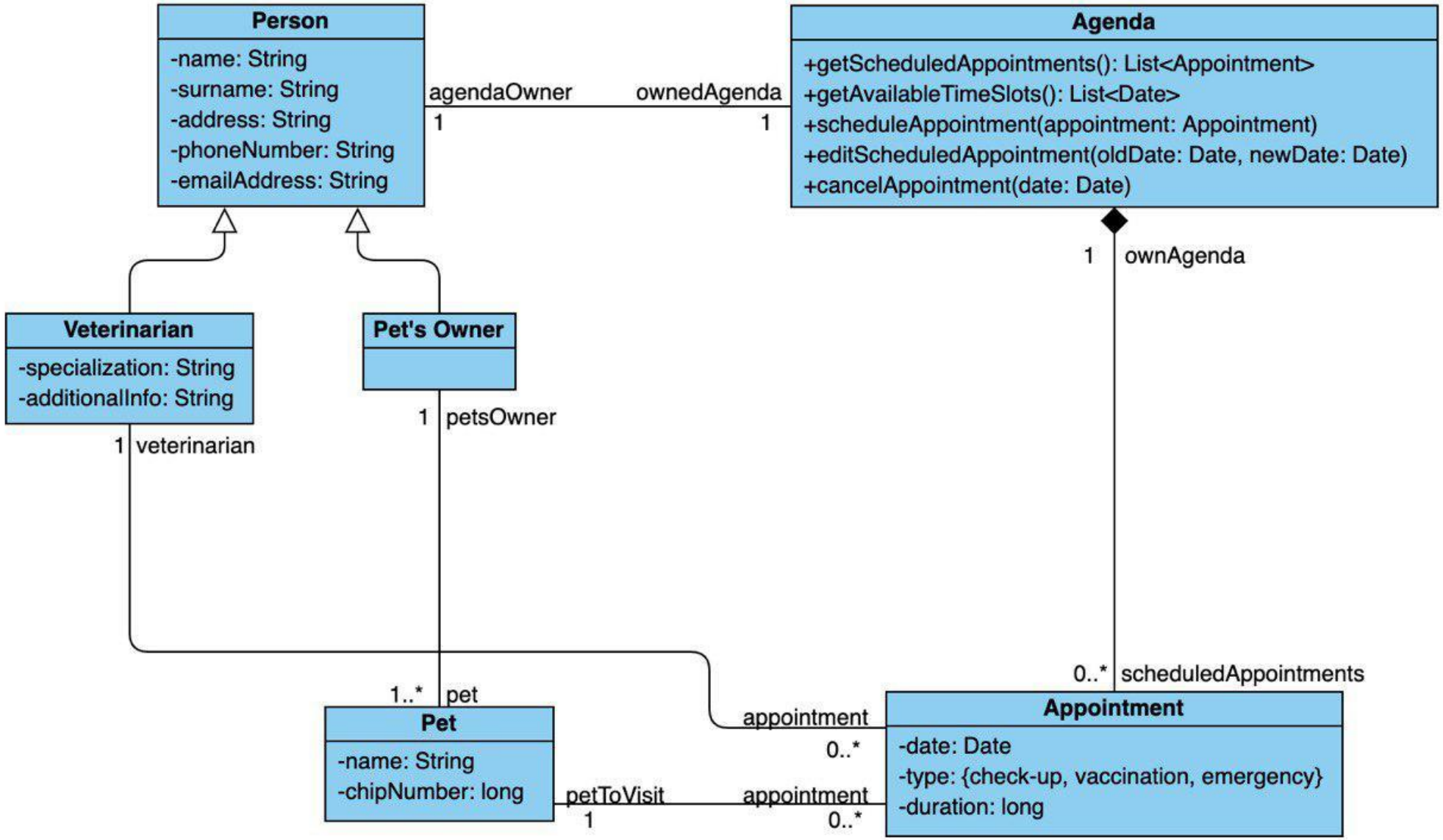

Periods & Ranges
Chart Display
Map Display
Pet's Owner
Monitor machine
Veterinarian
Calendar Display
Agenda
Database
Analog Devices
Pet
Retrieve and store data from pets,
Show data about pet's health status,
Show pet's position,
Generate, store and show periods & ranges,
Detect and notify alarms,
Generate, store and show appointments,
Show data about diets and therapy programs,
Show pet's owner stock info

Class Diagram


Ration
Person
Entry
Stock
Pet's Owner
Agenda
1..*
Sensor
Collar
Pet
Appointment
Program
0..*
Veterinarian
Diet
Therapy

The stock represents the set of provisions owned by the pet’s owner. It is composed by a list of entries, and each entry tracks the ration’s type and its amount. Each ration contains the specific information about macronutrients.


Ration
Person
Entry
Stock
Pet's Owner
Agenda
1..*
Sensor
Collar
Pet
Appointment
Alarm
Periods & Ranges
Program
0..*
Veterinarian
Diet
Therapy
InterestingCase
StockLimits
SecureArea
VitalFunctionLimits

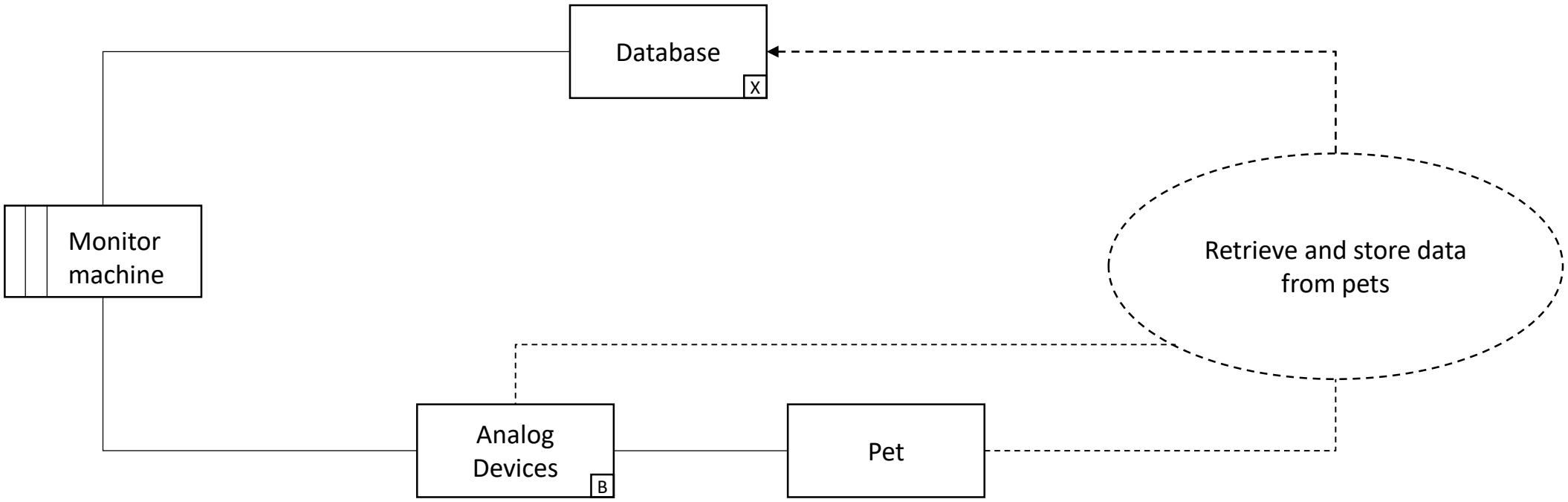

Person
-name: String
-surname: String
-address: String
-phoneNumber: String
-emailAddress: String
agendaOwner
1
ownedAgenda
1
Agenda
+getScheduledAppointments(): List<Appointment>
+getAvailableTimeSlots(): List<Date>
+scheduleAppointment(appointment: Appointment)
+editScheduledAppointment(oldDate: Date, newDate: Date)
+cancelAppointment(date: Date)
1
ownAgenda
Veterinarian
-specialization: String
-additionalInfo: String
Pet's Owner
1
petsOwner
1
veterinarian
1..*
pet
0..*
scheduledAppointments
Pet
-name: String
-chipNumber: long
appointment
0..*
Appointment
-date: Date
-type: {check-up, vaccination, emergency}
-duration: long
petToVisit
1
appointment
0..*

OCL

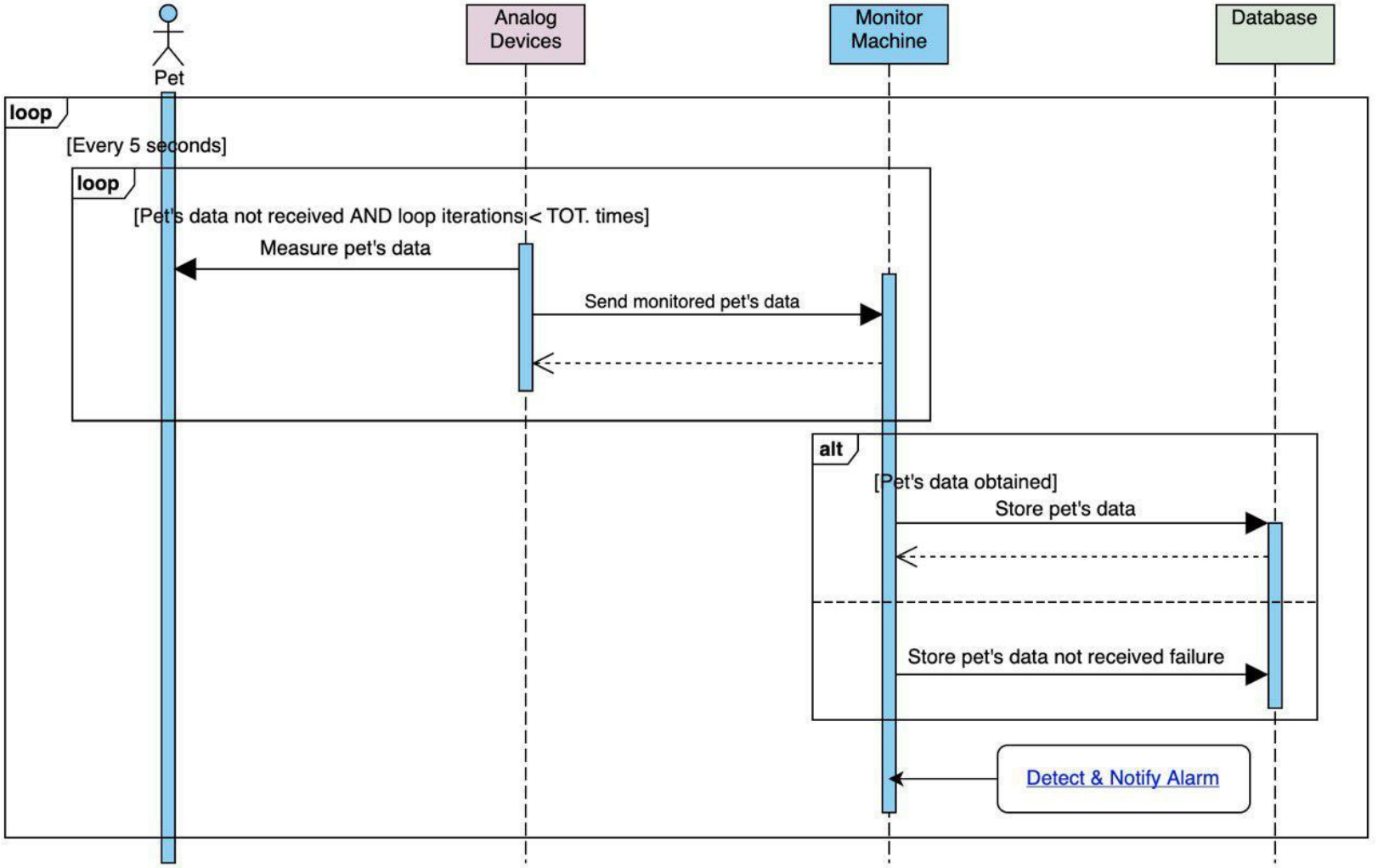
- context Agenda
  inv: self.scheduledAppointments -> forAll (a1, a2 | a1 <> a2 implies a1.date <> a2.date)
  inv: if (self.person.oclIsTypeOf(PetsOwner)) then
  self.scheduledAppointments -> forAll (a1 |
  a1.pet.petsOwner = self.person)
  else
  self.scheduledAppointments -> forAll (a1 |
  a1.veterinarian = self.person)
  endif
- context Agenda::getScheduledAppointments(): List<Appointment>
  pre: person -> notEmpty
  post: result = scheduledAppointments
- context Agenda::scheduleAppointment(appointment: Appointment)
  pre: not scheduledAppointments -> exists(a1 | a1.date = appointment.date)
  post: scheduledAppointments -> includes(appointment)
- context Agenda::editScheduledAppointment(
  oldDate: Date, newDate: Date)
  pre: oldDate <> newDate and scheduledAppointments -> exists(a1 | a1.date = oldDate)
  post: not scheduledAppointments -> exists(a1 | a1.date = oldDate)
  and scheduledAppointments -> exists(a1 |
  a1.date = newDate)
- context Agenda::cancelAppointment(date: Date)
  pre: person.oclIsTypeOf(Veterinarian) and
  scheduledAppointments -> exists(a1 | a1.date = date)
  post: not scheduledAppointments -> exists(a1 | a1.date = date)
- context Appointment
  inv: if(self.type = #emergency) then
  self.duration = 60
  else if(self.type = #vaccination) then
  self.duration = 20
  else
  self.duration = 40
  endif

Problem Frames

Retrieve and store data from pets

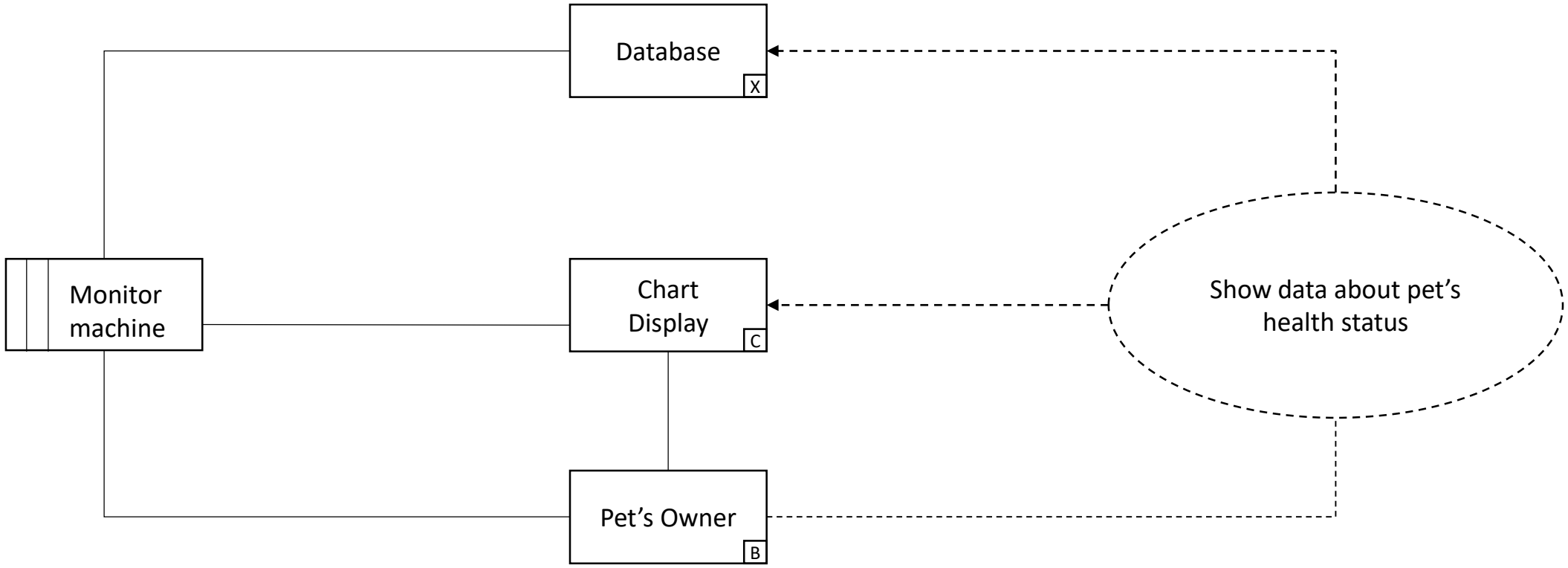


Retrieve and store data from pets



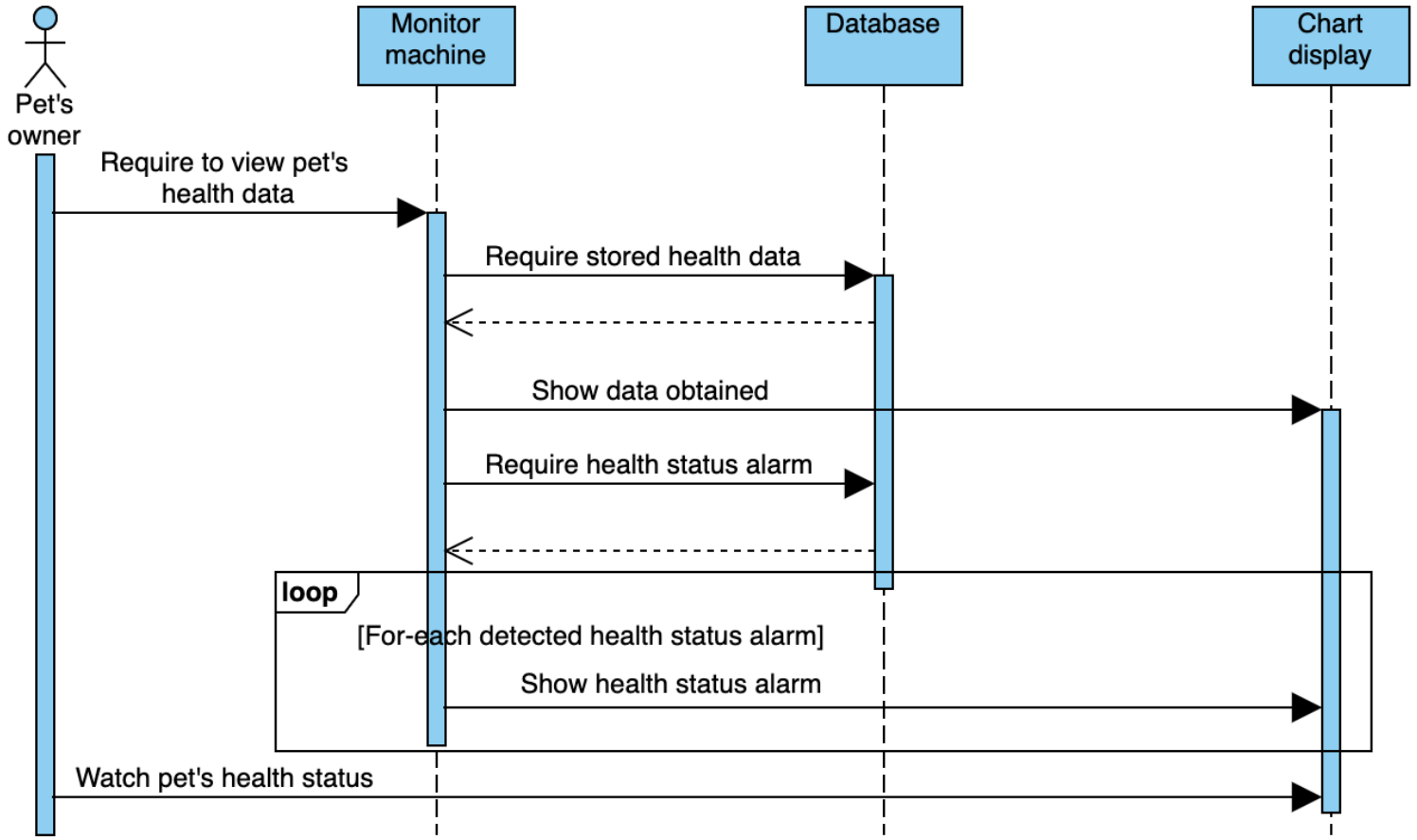
Pet's data are pet's vital functions and pet's position

Show data about pet’s health status

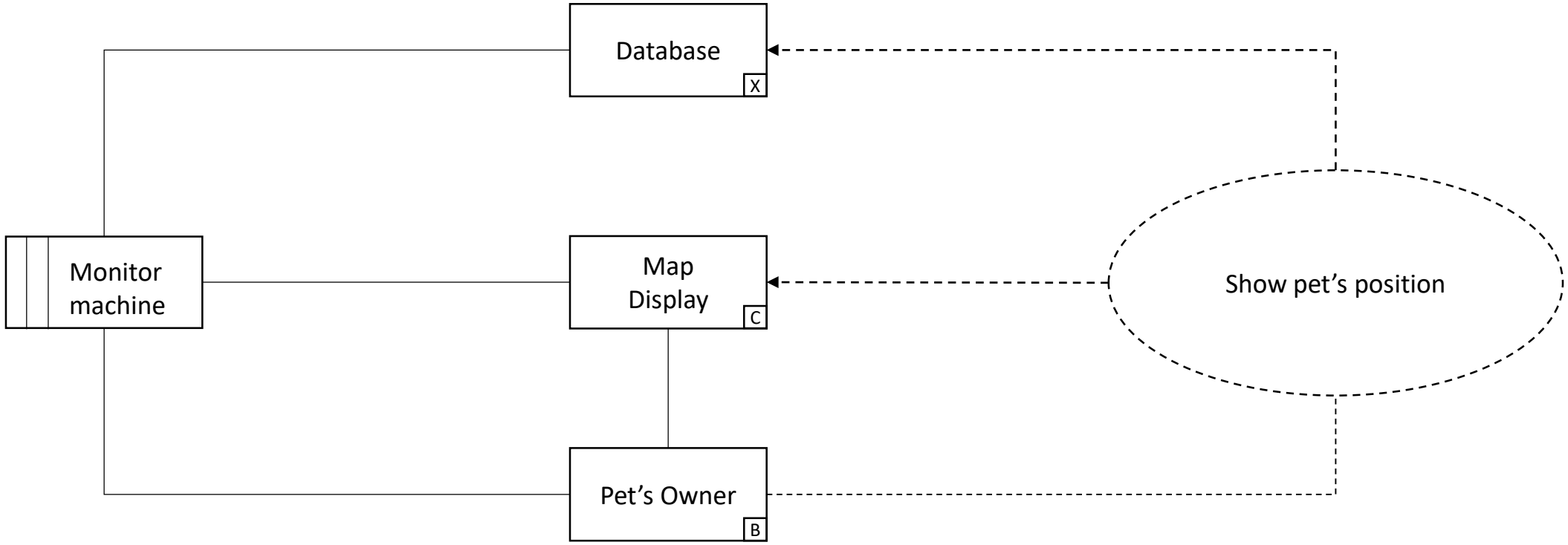


Enquiry

Show data about pet’s health status



Show pet’s position



Enquiry

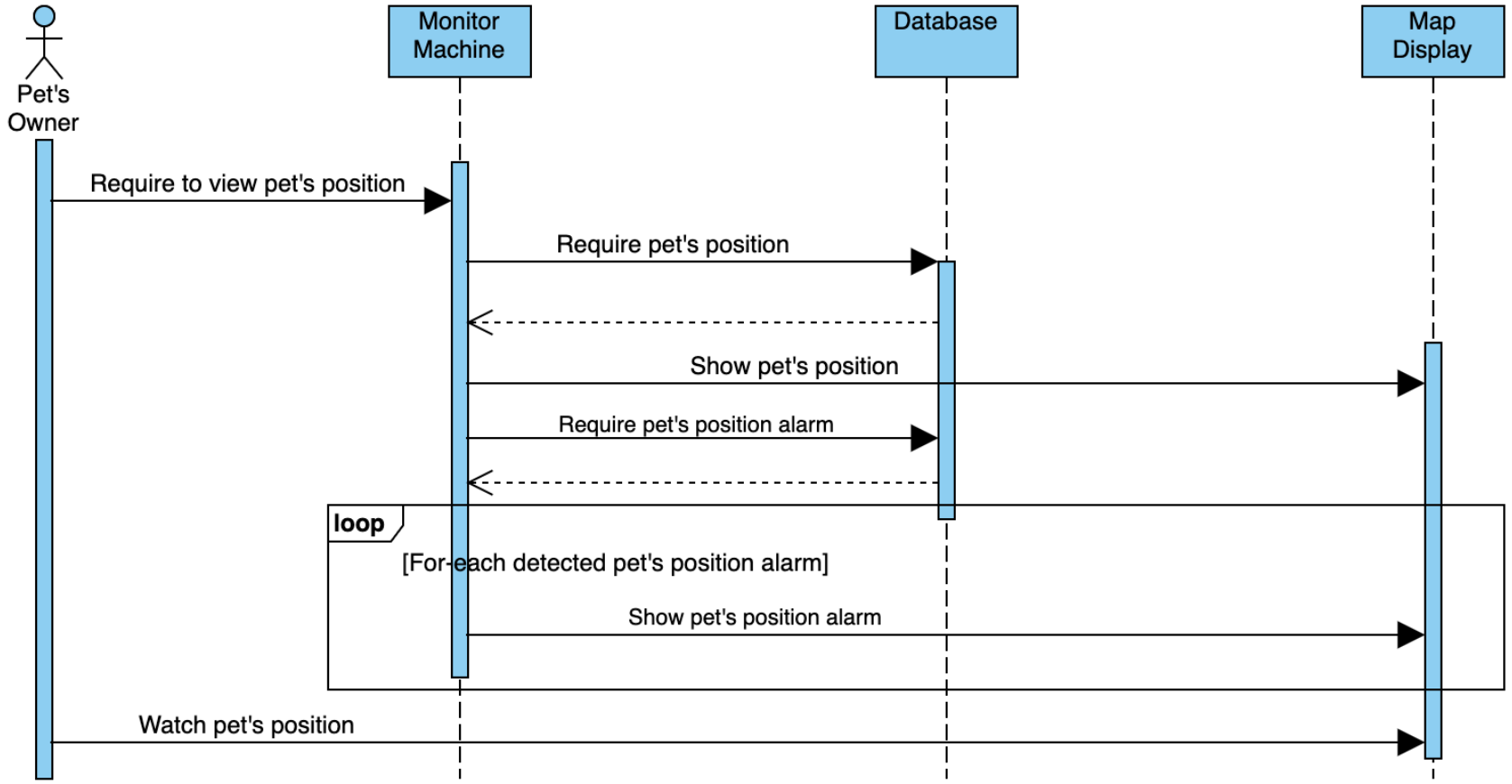
Show pet’s position

This problem frame is used when the pet’s owner wants to check where the pet is.

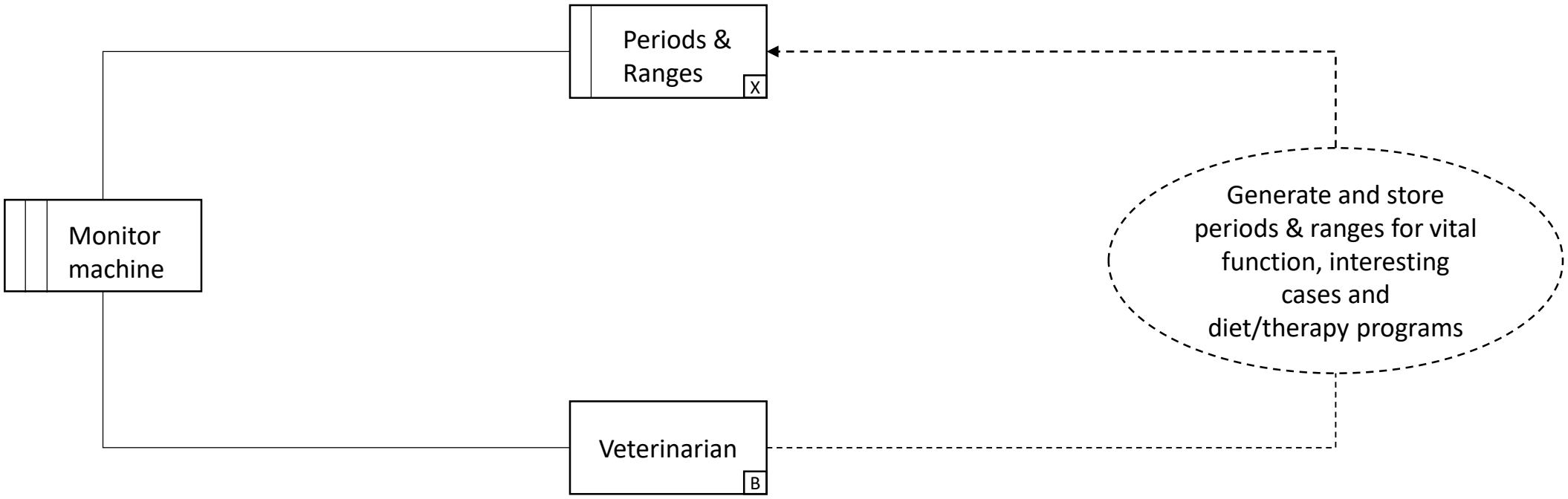
If there is an alarm, and so the pet is not inside the secure area limits, the monitor machine sends a notification to the pet’s owner and then he can check the position through this problem frame.

The alarm detection is not managed here.

Show pet’s position

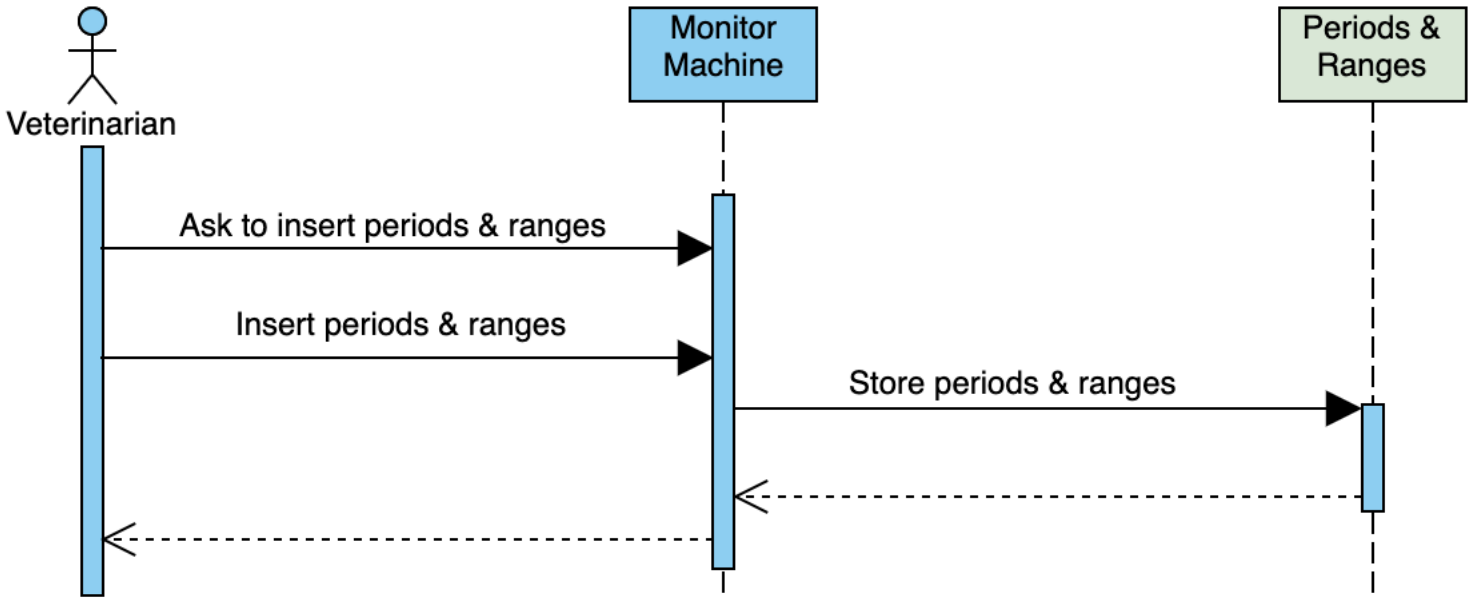


Generate and store periods & ranges

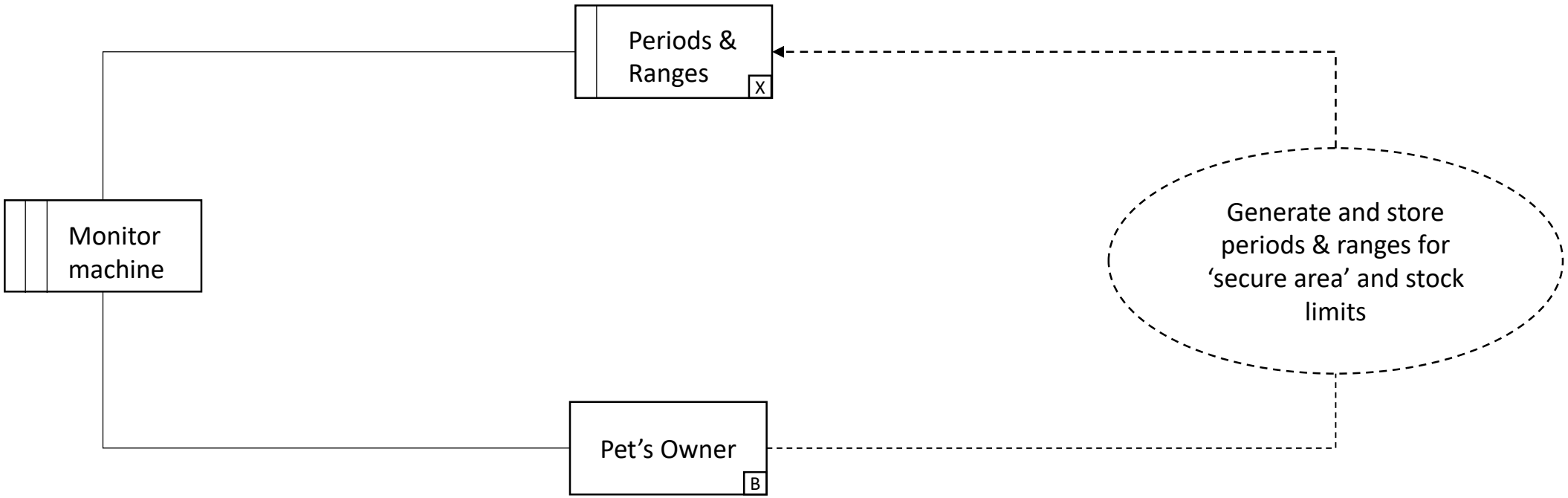


Workpiece

Generate and store periods & ranges

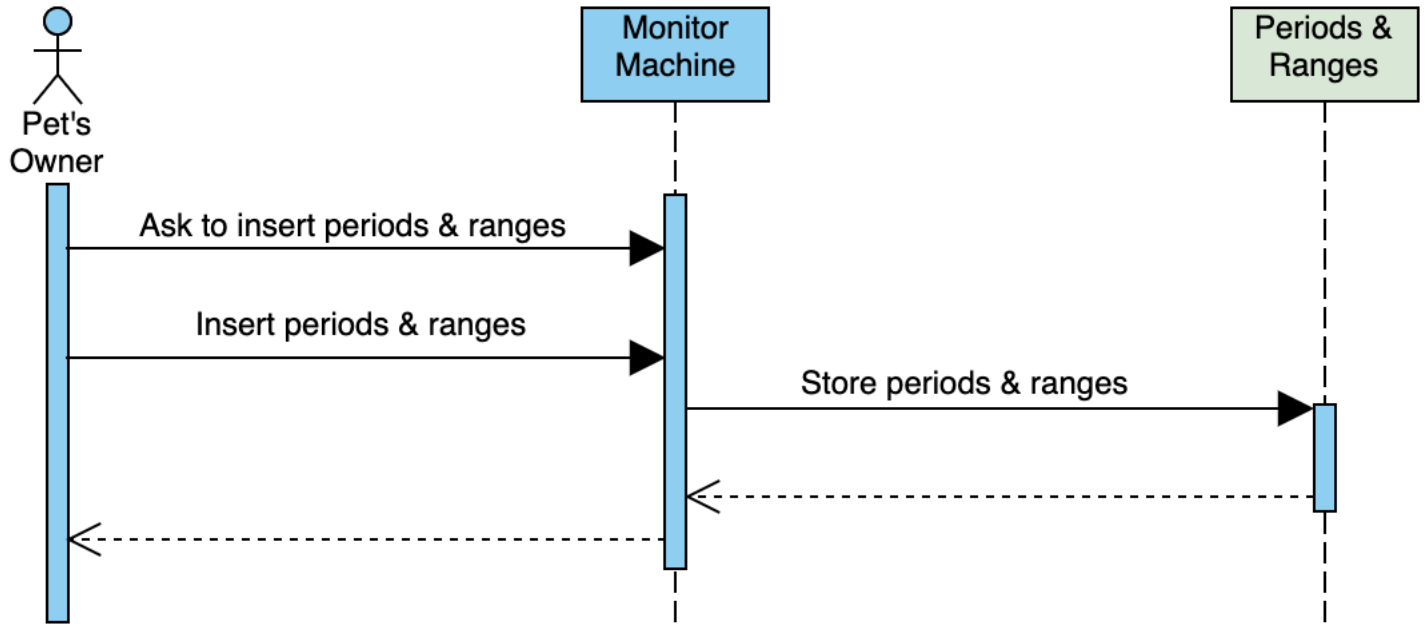


Generate and store periods & ranges

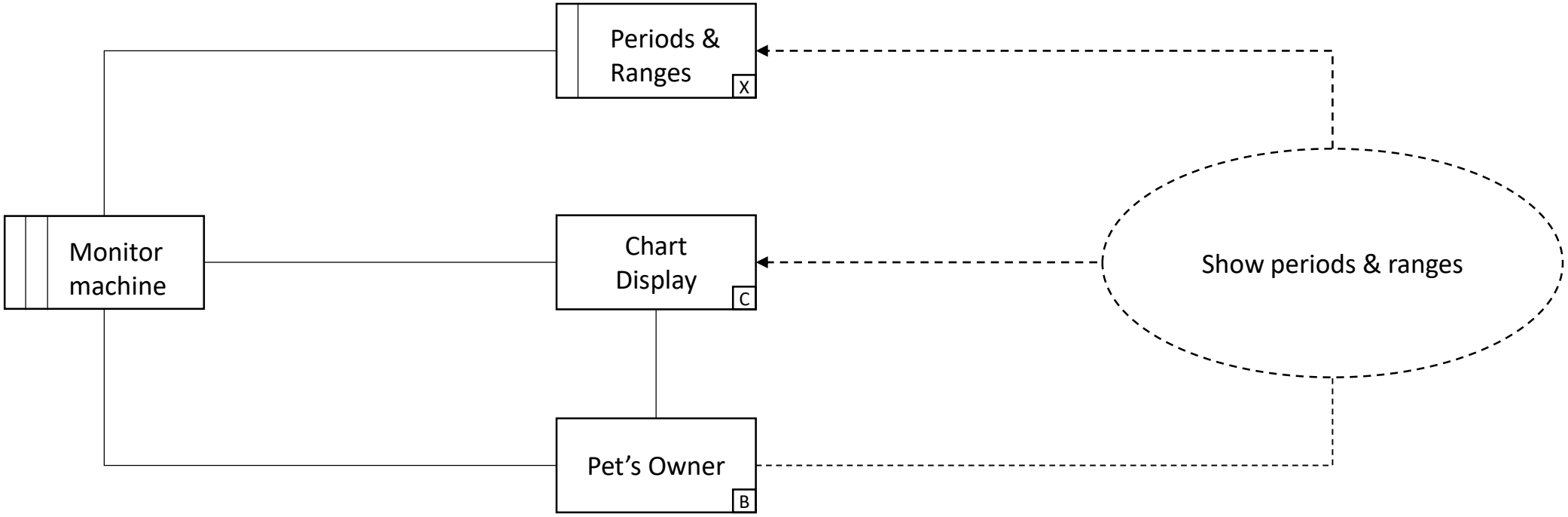


Workpiece

Generate and store periods & ranges

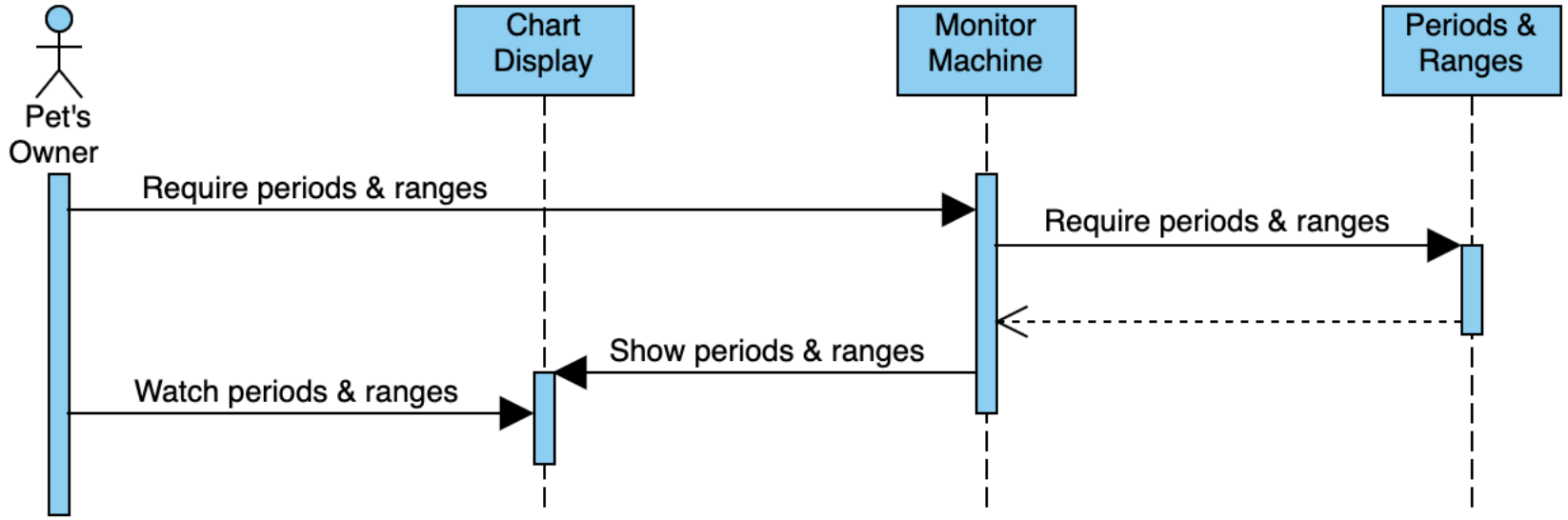


Show periods & ranges

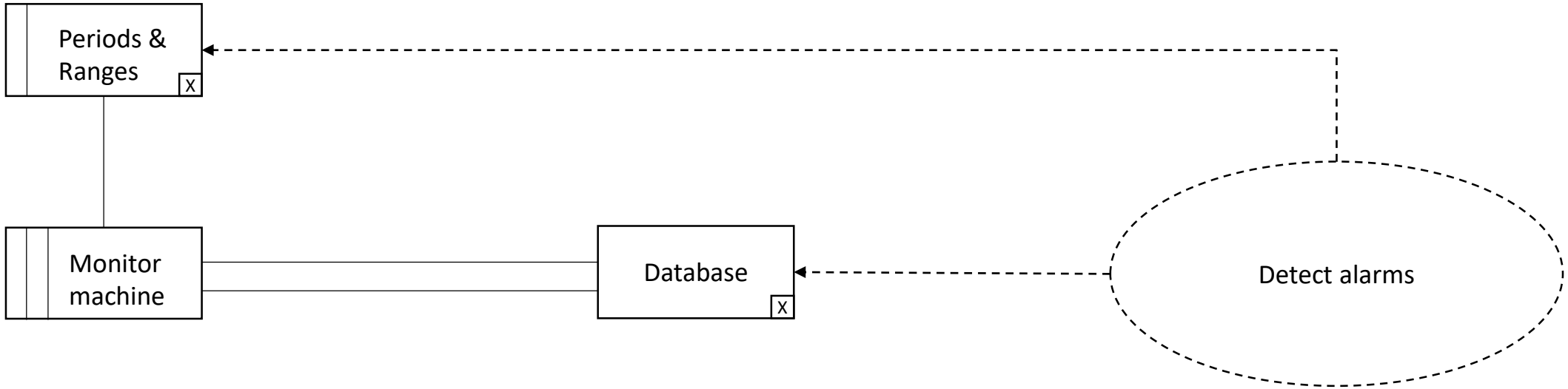


Enquiry

Show periods & ranges

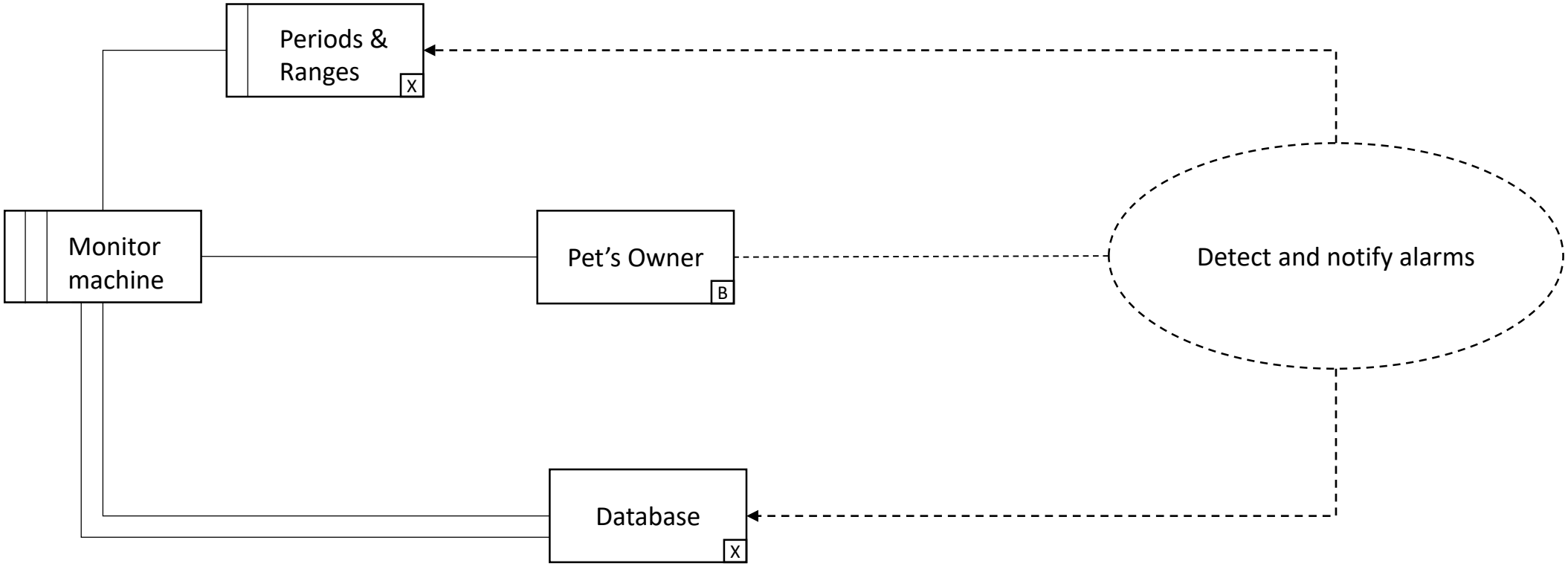


Detect alarms



Transformation

Detect and notify alarms



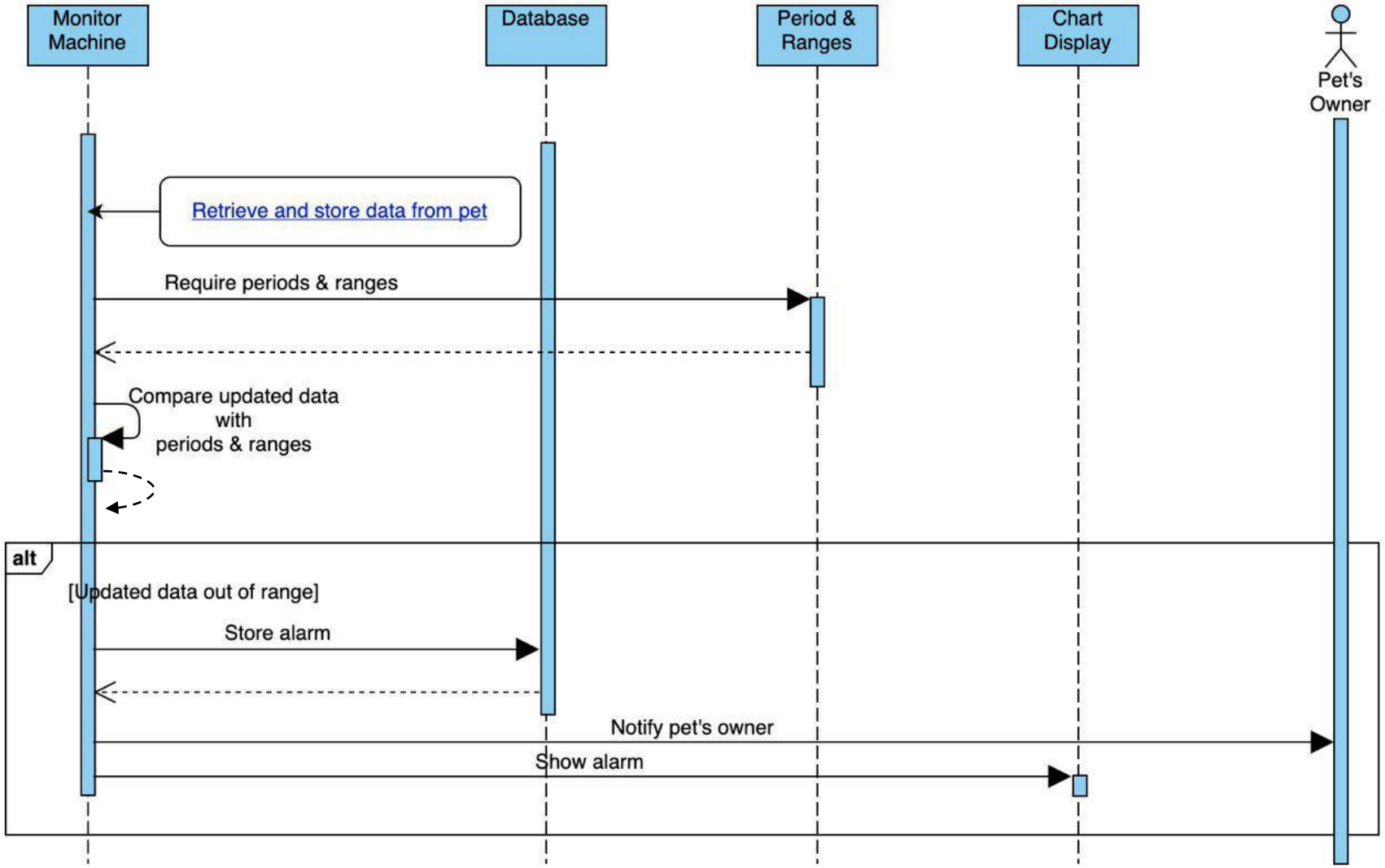
Transformation + Display

Detect and notify alarms

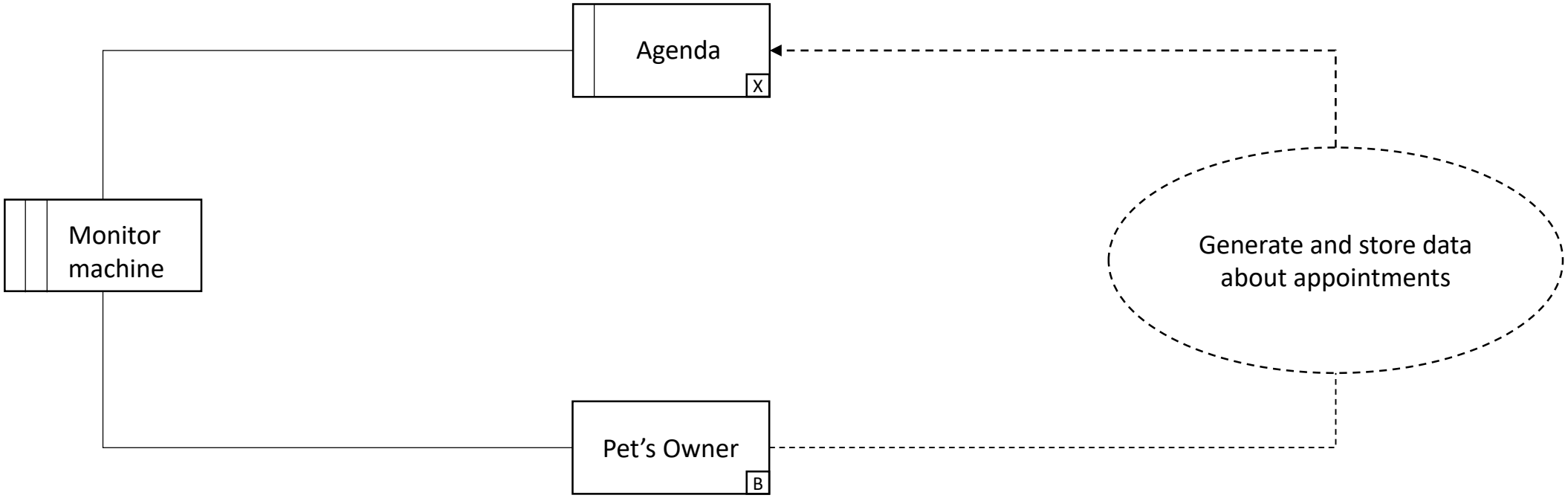
Each time the database receives new data, the monitor machine has to check if these data fall into the periods & ranges defined for that pet (and so if an alarm has to be notified to the pet's owner). If so, an alarm has to be created and stored into the DB.

Alarms are notified directly to the pet's owner, and they can be shown in the 'Show data about pet's health status' or 'Show pet's position' diagram.

Detect and notify alarms

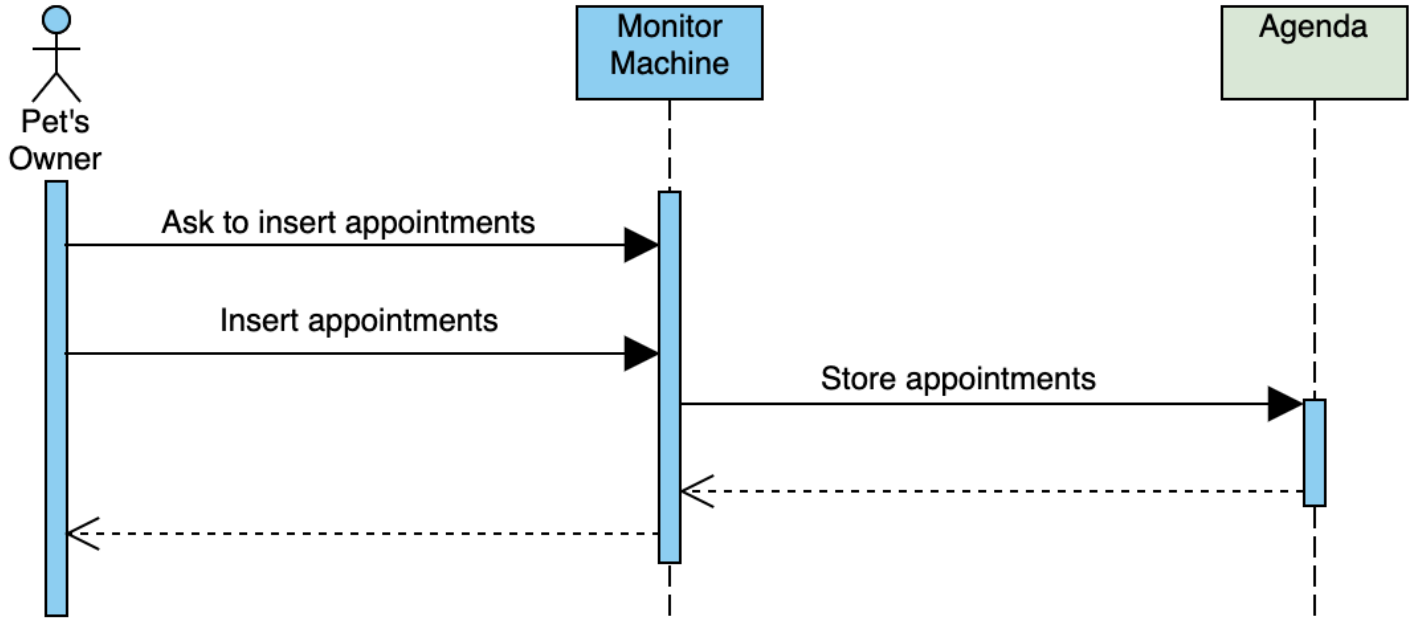


Generate and store data about appointments

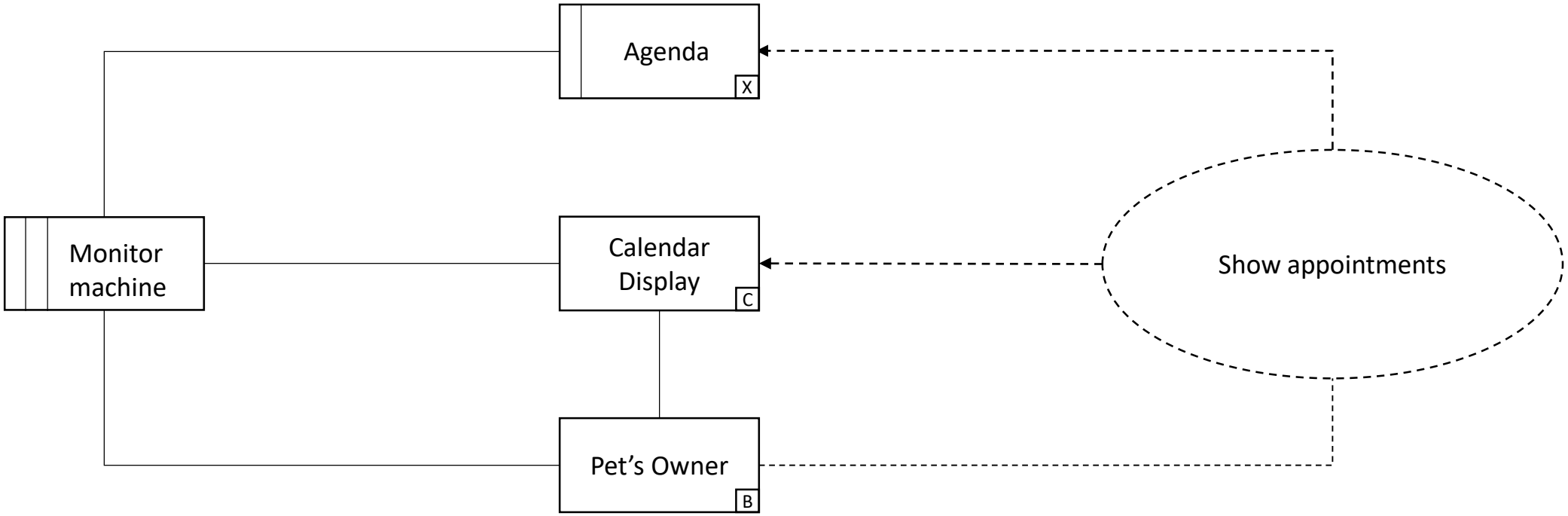


Workpiece

Generate and store data about appointments

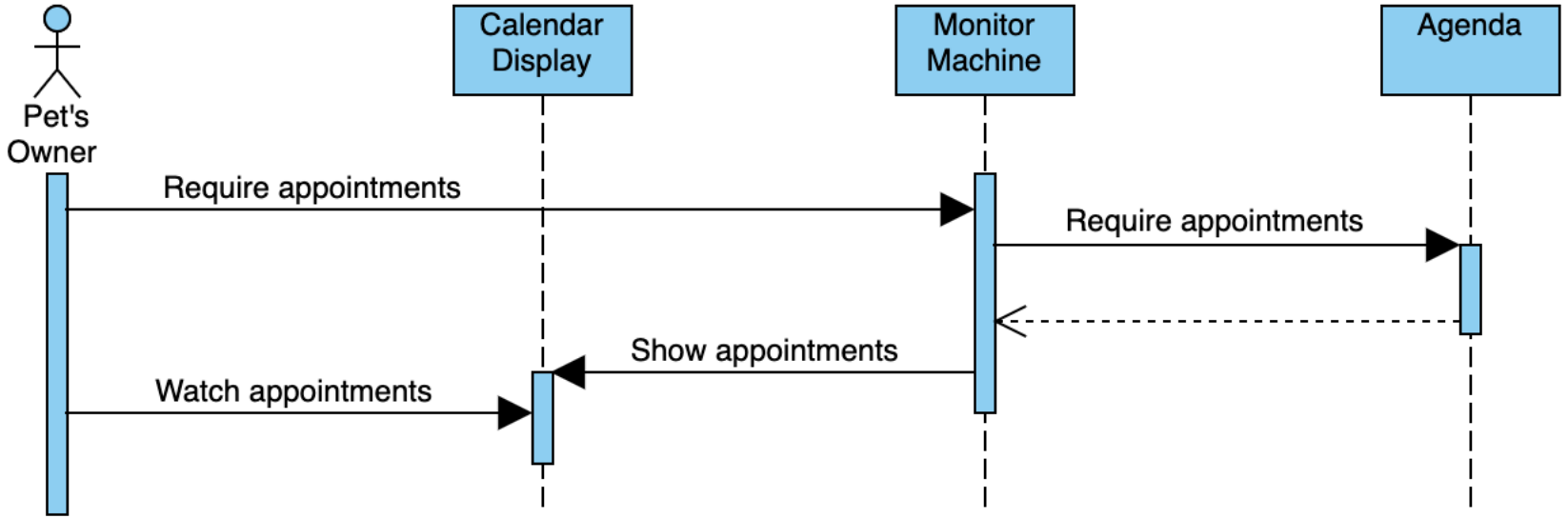


Show appointments

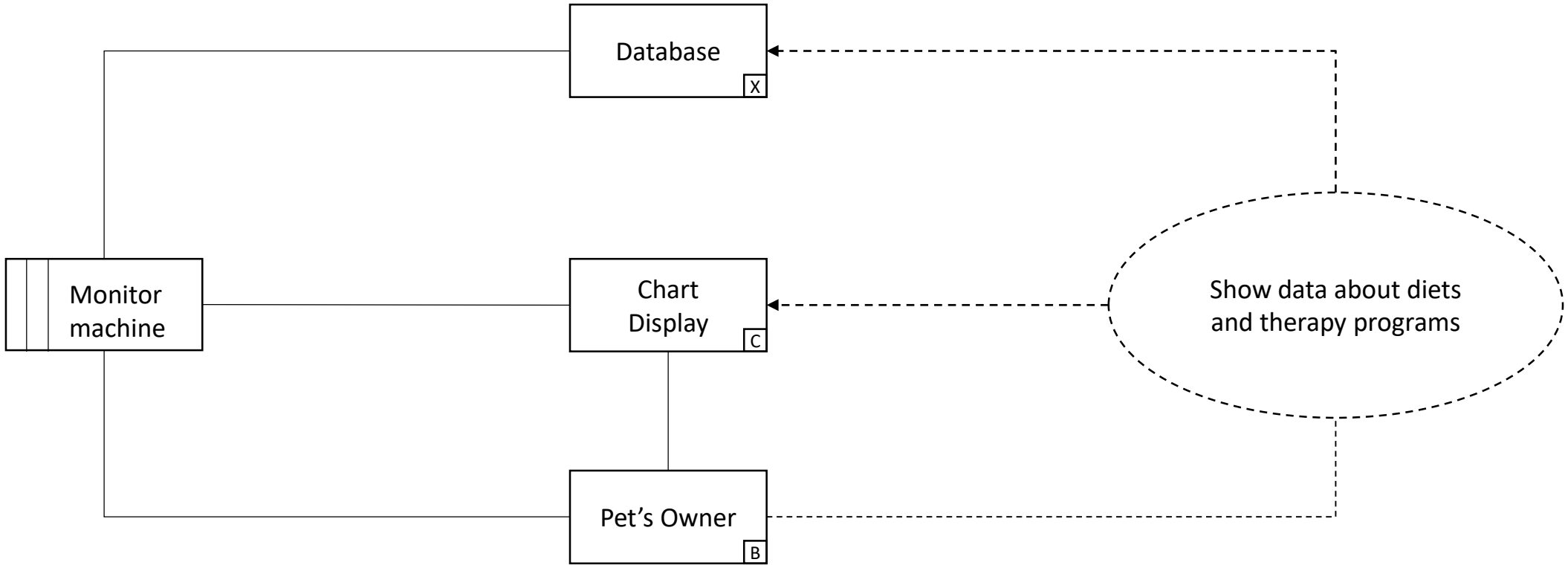


Enquiry

Show appointments

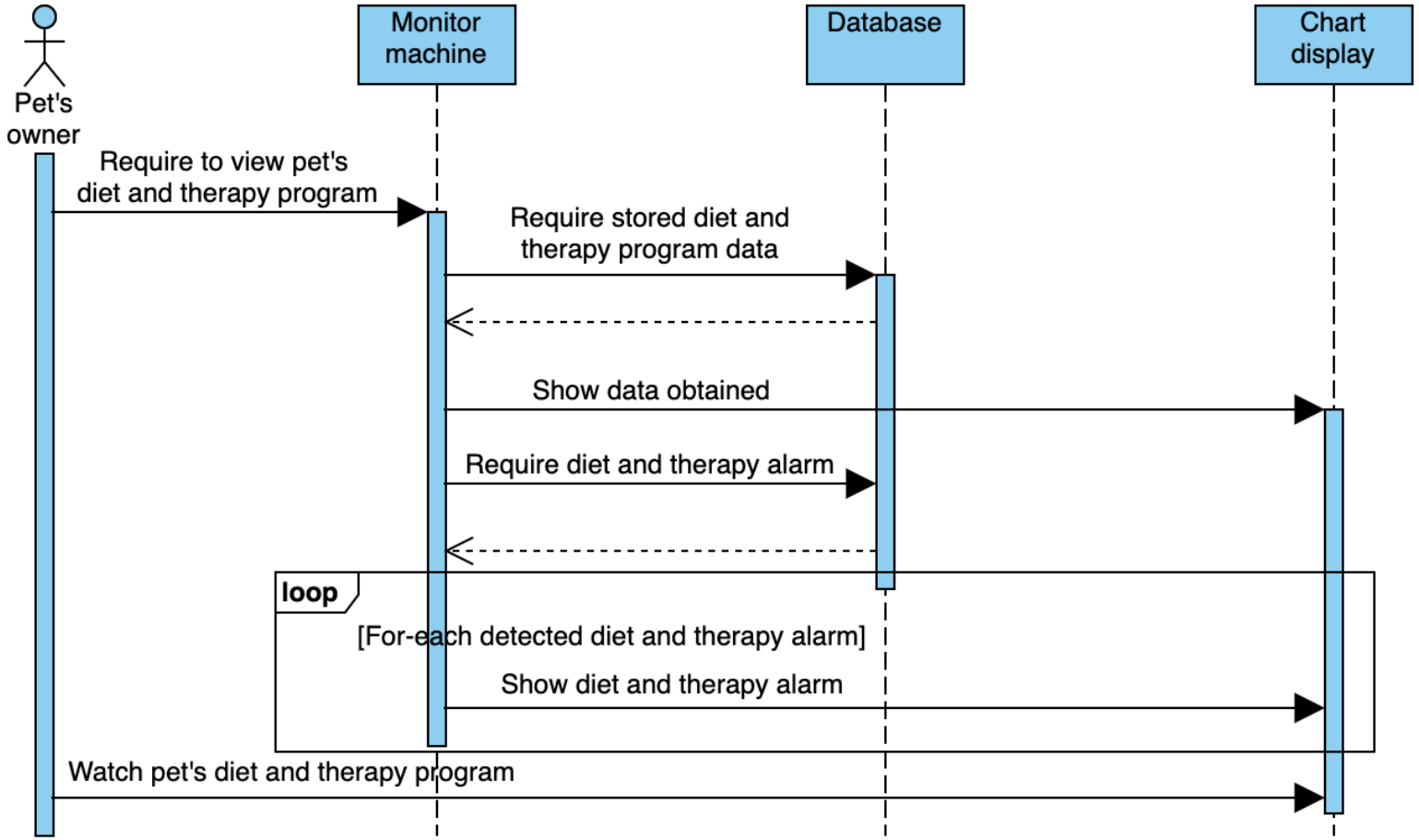


Show data about diets and therapy programs

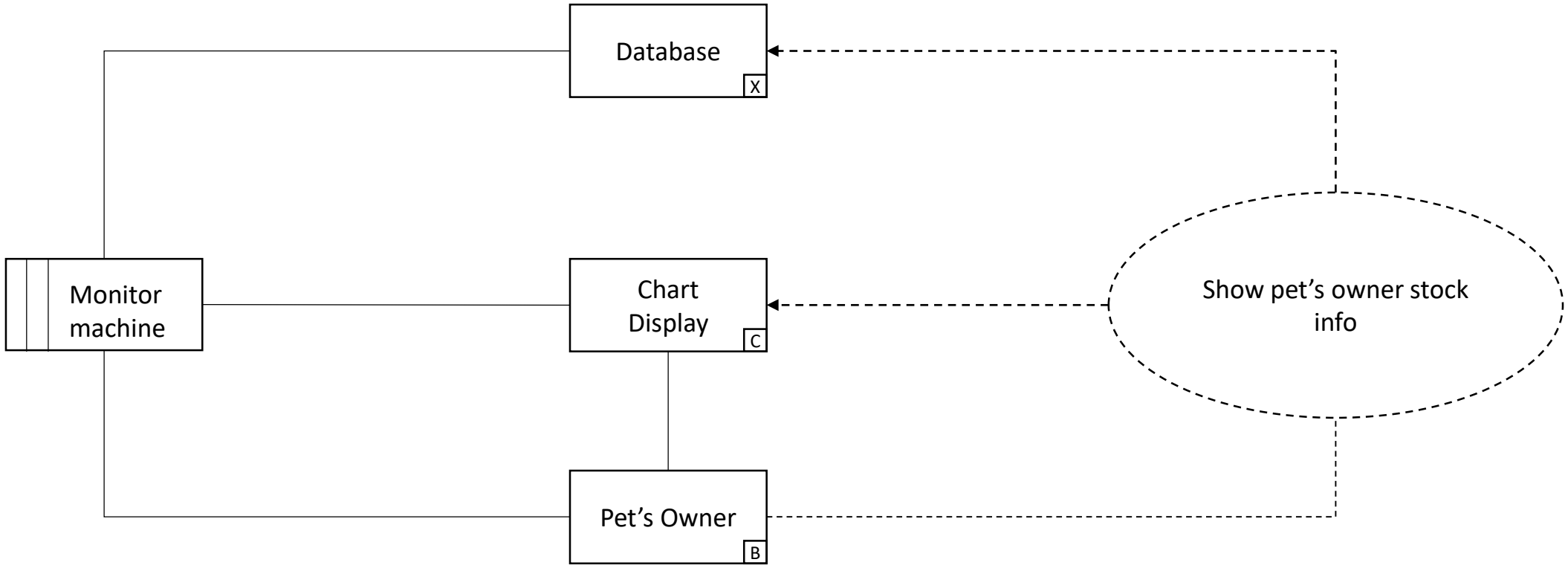


Enquiry

Show data about diets and therapy programs

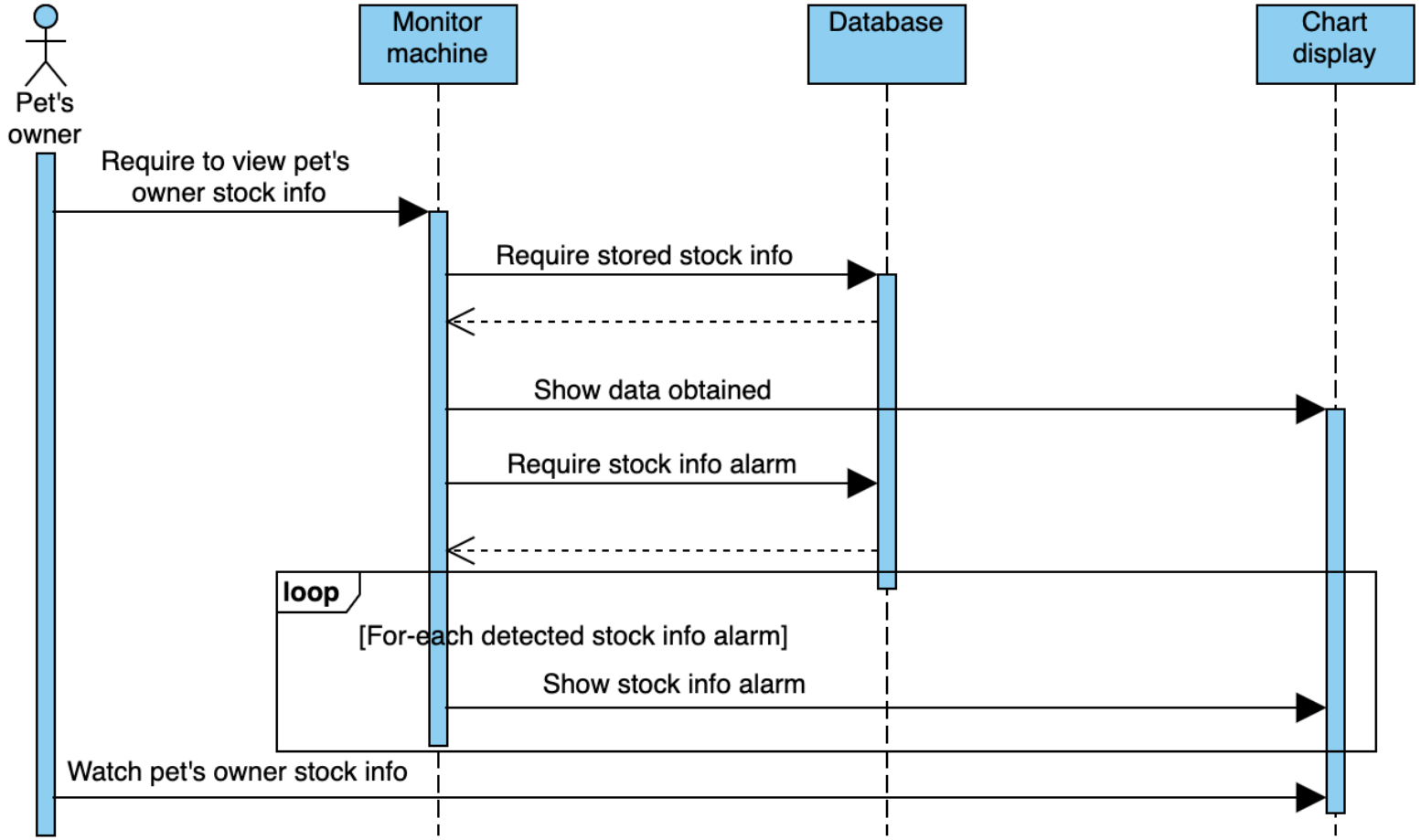


Show pet’s owner stock info



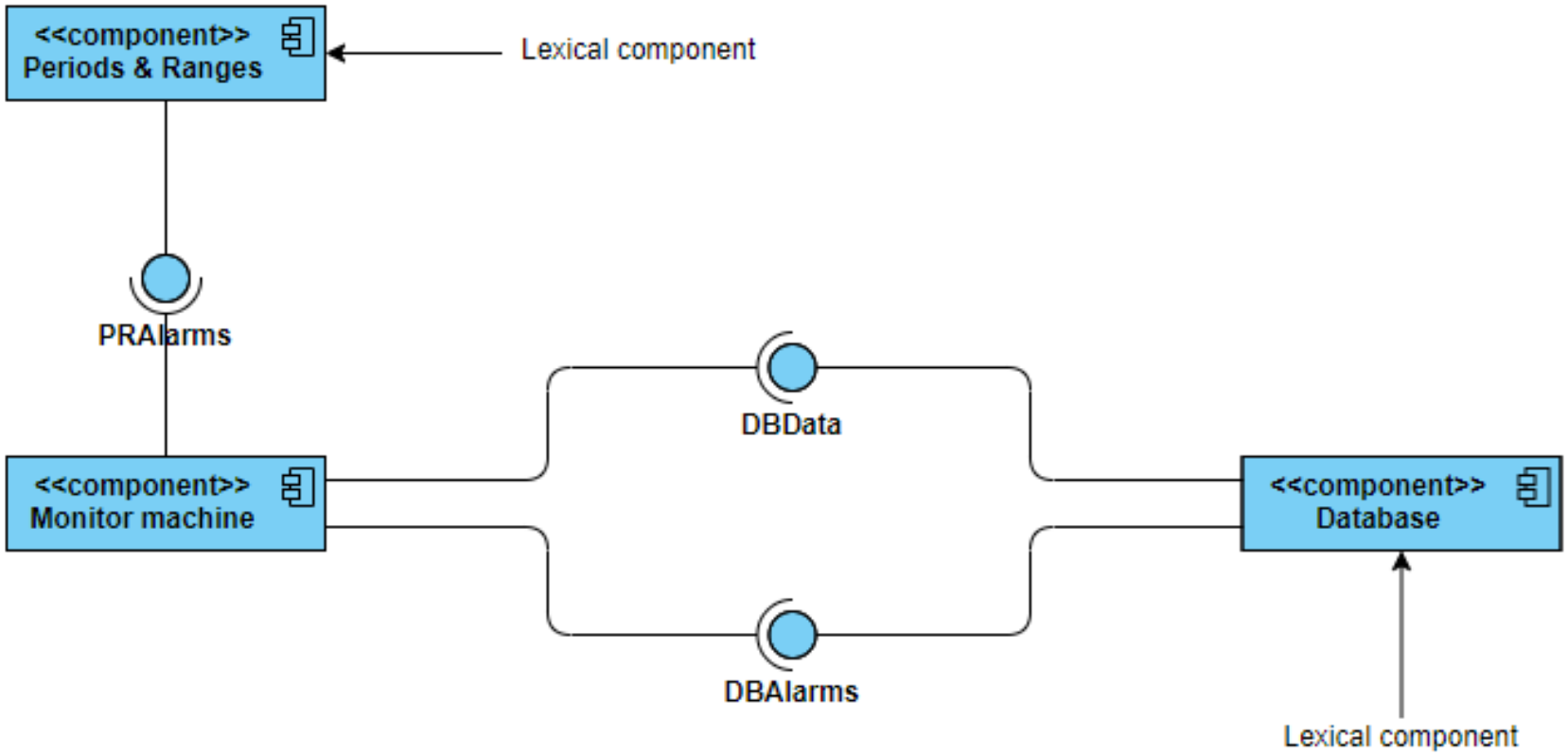
Enquiry

Show pet’s owner stock info

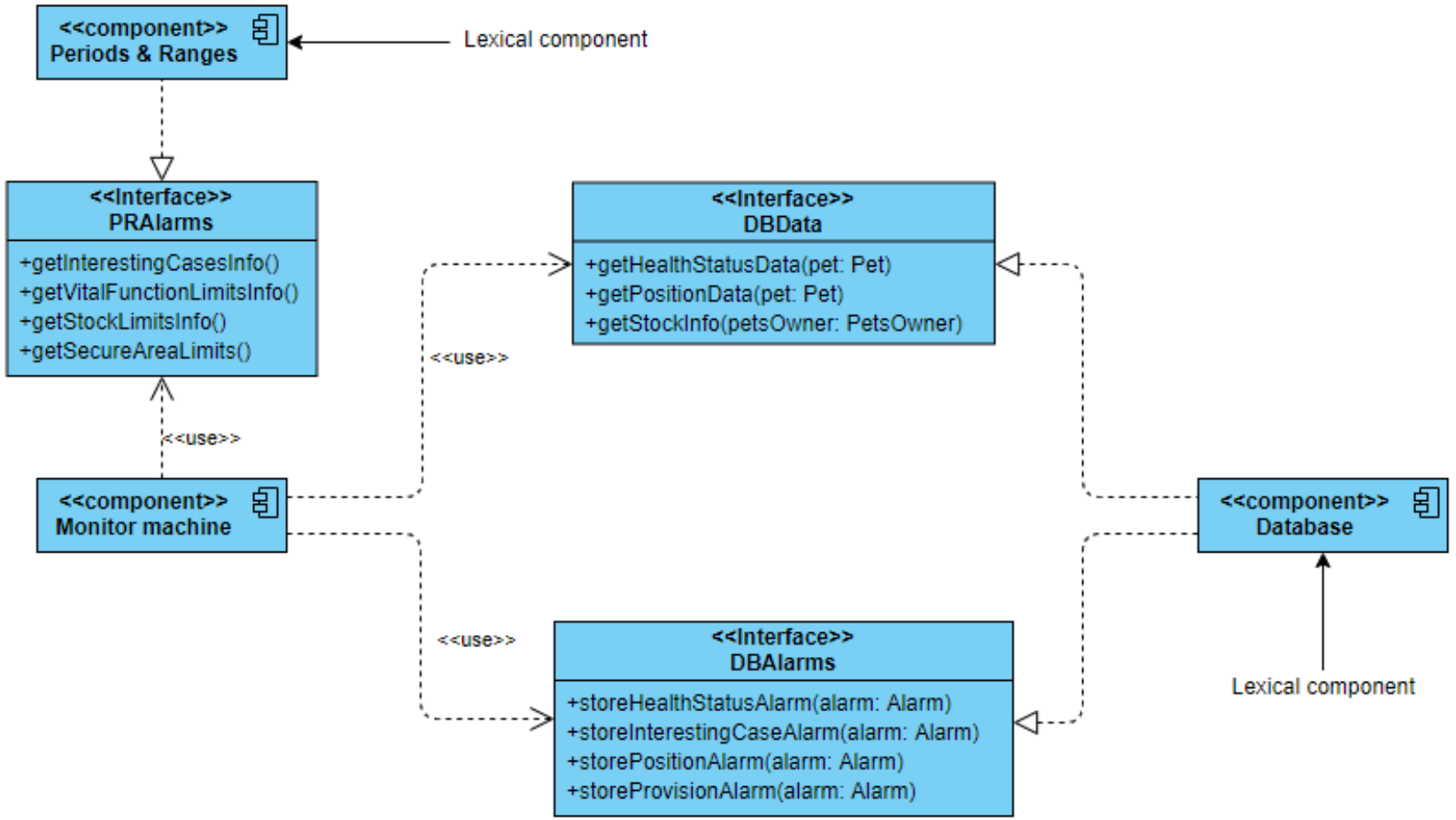


Component Diagrams

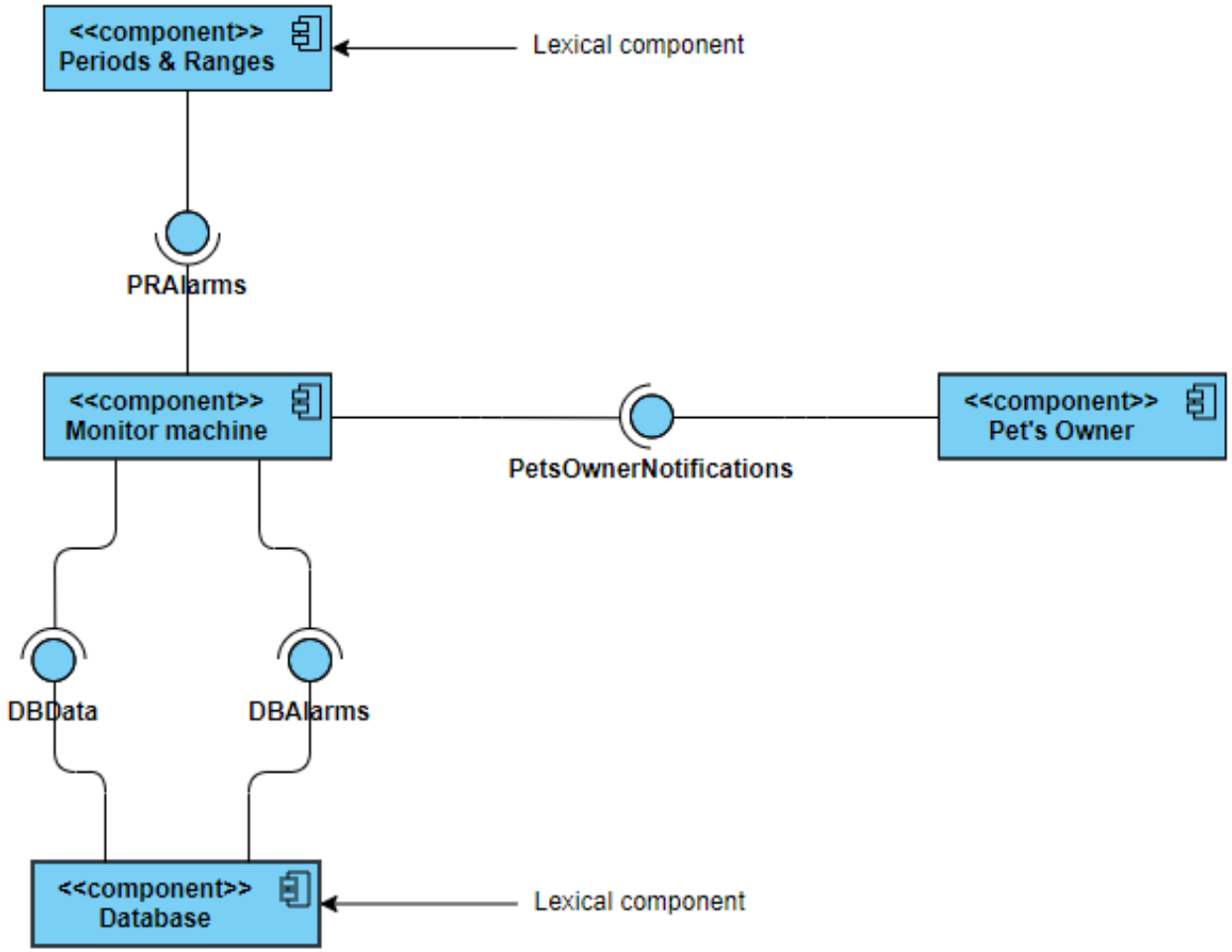
Detect alarms



Detect alarms



Detect and notify alarms



Detect and notify alarms