Context Diagram

Designations

Physical Structure
- **House**(h): h is a building
- **Room**(r): r is a room of the house
- **Window**(w): w is a window in an outside wall of the house
- **Door**(d): d a door in an outside wall of the house
- **HasDoor**(r,d): room r has a door d
- **HasTerminal**(r,t): room r has a terminal t

Persons/Roles
- **Authorities**(a): a are police or local security service authorities
- **Intruder**(i,h): i is a person forcefully trying to enter the house h
- **Tenant**(t,h): t is a person living in the house h
- **Neighbor**(n,h): n is a person living near house h

Input Devices
- **Sensor**(s): s is a device capable of detecting a problem
- **SmokeDetector**(s,r): s is a sensor capable of detecting smoke in room r
- **MotionDetector**(m,r): r is a sensor capable of detecting motion in a room r
- **WindowDetector**(s,w): s is a sensor capable of detecting an open window w
- **DoorDetector**(s,d): s is a sensor capable of detecting an open door d
- **ExternalMonitoringSystem**(e): e is a system monitoring other state

Input/Output Devices
- **Terminal**(t): t is the device providing a user interface to the security system

Output Devices
- **AlarmSystem**(a): a is the system that propagates alarms (to sirens/horns/lights and the authorities)

Dynamic Domain Fundamental Definitions
- **Event**(e): e is an atomic event
- **Earlier**(e,f): event e occurs before event f
Interval(v)  v is an interval of time  
Duration(d,v)  d is the duration (in seconds) of interval v  
Begins(e,v)  event e begins interval v  
Ends(e,v)  event e ends interval v  

Specific Events  
Motion(e,r)  in event e, some movement happens in room r  
Open(e,x)  in event e, a door or window x opens  
Close(e,x)  in event e, a door or window x closes  
Signal(e,s)  in event e, sensor s signals a problem  
LightMotionIndicator(e) in event e, light the motion indicator on every terminal  
DimMotionIndicator(e) in event e, dim (switch off) the motion indicator on every terminal  
LightWindowIndicator(e) in event e, light the open window indicator on every terminal  
DimWindowIndicator(e) in event e, dim (switch off) the open window indicator on every terminal  
LightDoorIndicator(e) in event e, light the open door indicator on every terminal  
DimDoorIndicator(e) in event e, dim (switch off) the open door indicator on every terminal  
SwitchOnAlarm(e) in event e, the alarm system switches on an acoustic and/or visual feedback  
SwitchOffAlarm(e) in event e, the alarm system switches off the alarm  
SendNotification(e) in event e, the alarm system sends a notification about an alarm to the authorities  
Authenticates(e) in event e, someone correctly authenticates himself to gain access  
PressAway(e) in event e, someone presses the "away" button  
PressHome(e) in event e, someone presses the "home" button  
PressOff(e) in event e, someone presses the "off" button  

System States  
Away(v) in interval v, the system is in "away" mode  
Home(v) in interval v, the system is in "home" mode  
Off(v) in interval v, the system is in "off" mode  

Definitions  
Dynamic Domain Definitions  
InitialInterval(v) : Interval(v) and not exists e: Begins(e,v)  
FinalInterval(v) : Interval(v) and not exists e: Ends(e,v)  
Includes(v,e) : Interval(v) and exists e,i,f: Begins(i,v) and Ends(f,v)  
Duration(d,e,f) : Duration(d,v) and Begins(e,v) and Ends(f,v)  

Other Definitions  
MotionDetected(e,s,r) : MotionDetector(s,r) and Signals(e,s)  
OpenWindowDetected(e,s,w) : WindowDetector(s,w) and Signals(e,s)  
OpenDoorDetected(e,s,d) : DoorDetector(s,d) and Signals(e,s)  

Refutable Descriptions  
Static: Rooms, windows and doors are protected  
Room(r) => MotionDetector(s,r)  
Window(w) => WindowDetector(s,w)  
Door(d) => DoorDetector(s,d)  

Static: Easy to read the current state of the system & Authentication is straightforward  
Room(r) and HasDoor(r) and => exists t: Terminal(t) and HasTerminal(r,t)  

Dynamic: Sensors actually detect problems  
Room(r) and Motion(e,r) and MotionDetector(s,r) => MotionDetected(e,s,r)  
Window(w) and Open(e,w) and WindowDetector(s,w) => OpenWindowDetected(e,s,w)
Door(d) and Open(e,d) and DoorDetector(s,d) => OpenDoorDetected(s,d)

**Dynamic: Easy to read the current state of the system**

MotionDetected(e,s,r) and Away(v) and Includes(v,e) => LightMotionIndicator(e)
MotionDetected(e,s,r) and Home(v) and Includes(v,e) => LightMotionIndicator(e)
OpenWindowDetected(e,s,w) and Away(v) and Includes(v,e) => LightWindowIndicator(e)
OpenWindowDetected(e,s,w) and Home(v) and Includes(v,e) => LightWindowIndicator(e)
OpenDoorDetected(s,d) and Away(v) and Includes(v,e) => LightDoorIndicator(e)
OpenDoorDetected(s,d) and Home(v) and Includes(v,e) => LightDoorIndicator(e)

**Dynamic: Detected problems actually initiate alarms**

MotionDetected(e,s,r) and Away(v) and Includes(v,e) => SwitchOnAlarm(f) and Duration(10,e,f)
OpenWindowDetected(e,s,w) and Away(v) and Includes(v,e) => SwitchOnAlarm(f) and Duration(10,e,f)
OpenWindowDetected(e,s,w) and Home(v) and Includes(v,e) => SwitchOnAlarm(f) and Duration(10,e,f)
OpenDoorDetected(s,d) and Away(v) and Includes(v,e) => SwitchOnAlarm(f) and Duration(10,e,f)
OpenDoorDetected(s,d) and Home(v) and Includes(v,e) => SwitchOnAlarm(f) and Duration(10,e,f)

**Initiated Alarms actually lead to authority notifications**

SwitchOnAlarm(e) => SendNotification(e,u)

**Switching between modes**

Off(v) and Authenticates(e) and PressHome(e) => Home(w) and Begins(e,w) and Ends(e,v)
...dito for every other operating mode change
Off(v) and Begins(e,v) => DimMotionIndicator(e) and DimWindowIndicator(e) and DimDoorIndicator(e)
Off(v) and Begins(e,v) => SwitchOffAlarm(e)

**Informal Refutable Descriptions**

Authenticates(t) must be possible within 20 seconds from entering any door.
A sensor must be installable in 5 minutes.
Every terminal must have a brief user manual attached.

**Rough Sketches**

- Various levels of security (away and at home). [*OK, formalized by the three security levels above*]
- Windows, doors, and rooms are protected. [*OK, formalized above*]
- Sensors can be installed without requiring major modification to the home’s infrastructure (e.g. prefer wireless devices over running cable through walls). [*Partially addressed with the informal requirement above*]
- Easy to read the current state of the system. [*Partially addressed by the requirement that alarms have to be indicated on the terminals, and that every room with a room has to have a terminal*]
- If an alarm is activated there shall be given enough information to respond appropriately. [*Partially addressed by the requirement that alarms have to be indicated on the terminals*]
- If an alarm goes off, the authorities are automatically notified. [*OK, formalized above*]
- The system is authenticated such that only tenants can modify the system’s operating mode. [*Partially addressed by using the Authenticates(e) predicate above*]
- Authentication is straightforward (so as not to take too much time). [*Partially addressed by the requirement that every room with a door must have a terminal, and with the informal refutable description above*]
- The security system can integrate other monitoring systems, such as a home’s smoke alarms. [*not addressed in refutable descriptions yet, only listed here for future specification in a fashion similar to the existing sensors*]
Remarks:
I started out using a rigorous predicate logic approach. But I found that doing that forces me to define way too many details, especially in the dynamic part. I think that starting with the rough sketch, which, in my opinion corresponds to what you provided in the problem statement, would have been better.

In general I think that creating the formal predicate logic definitions and refutable descriptions leads to a structure which is very similar to a procedural system, but with much more details. I think that abstracting details away is much harder to do in this predicate logic structure than in an object-oriented structure. The only kind of abstraction provided by predicate logic actually is the substitution of predicates defined in terms of other predicates. In OO, we would have the more powerful means of composition and inheritance, but also already predefined abstractions for time (the order of program execution) and the same abstraction we have in predicate logic, functional decomposition (method calls). And we would also have powerful means of encapsulation, which are completely absent in predicate logic (in this respect, predicate logic seems to be the worst possible choice, since it's universal and existential quantifier break any kind of encapsulation). I remember from the algebraic specification languages (like Larch and JML), that using an OO approach on the outside (the relationships between subdomains/classes), and only using predicate logic to specify the internal workings of a class (like for class/object invariants and method pre- and postconditions) can be very beneficial.