

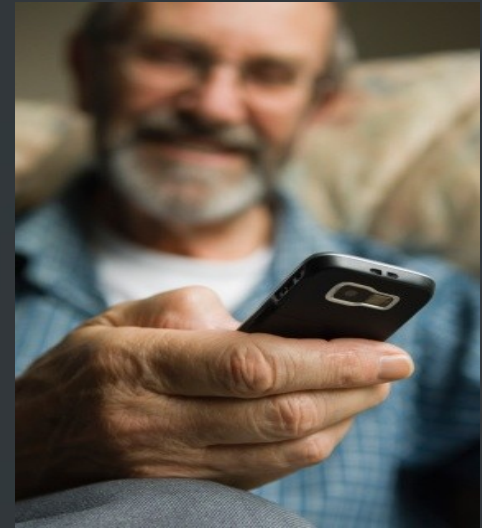
Model and In/outputs



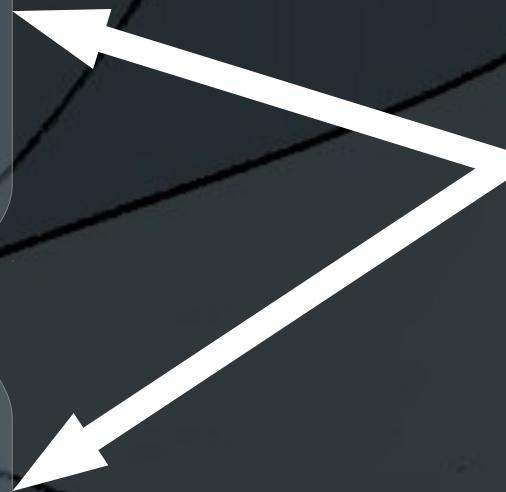
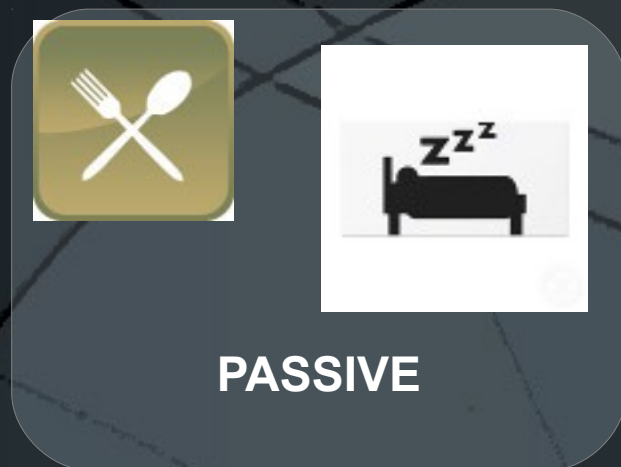
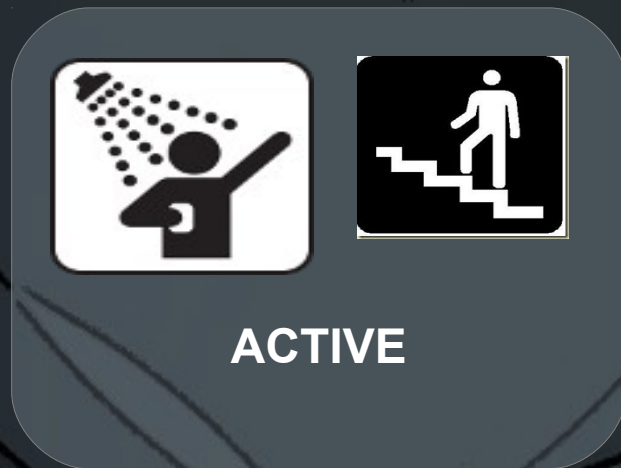
- What to model ?
 - To infer the inhabitant's health condition.
- Inputs ?
 - Context (Location, activity, time, heart beat of inhabitant, ...)
- Output ?
 - Show if health condition is normal, low risk or high risk.

Classes (1)

- We have a person who lives in the house (inhabitant)



Classes(2)



User has some actions in the house that are divided into 2 categories; active actions and passive actions.

Classes (3)



Locations



User is always in a location in the house such as kitchen, bedroom, bathroom, living room and so on. This is another context to be used while inferring the inhabitant's health condition.

Classes (4)

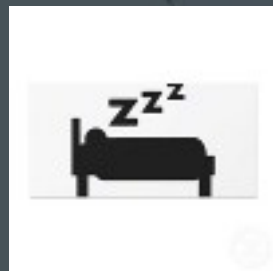


- Sensors : In this scenario, a dummy heart rate sensor is implemented to function as if it is real. It generates Bpm values (each 3 sec) from 75 to 180 so that this value can help the system determine the inhabitant's current health condition.
- The sensor class has other subclasses rather than heart rate sensor such as blood sugar sensor, temperature sensor. (They can be studied as future work)

Scenario Overview



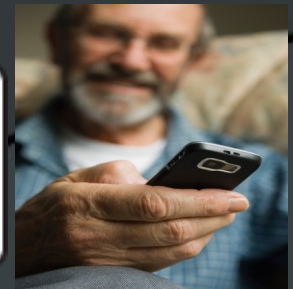
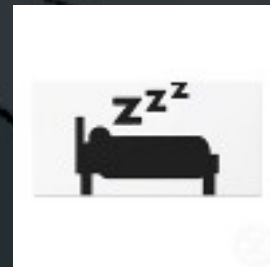
Any risk ?



Normal ?

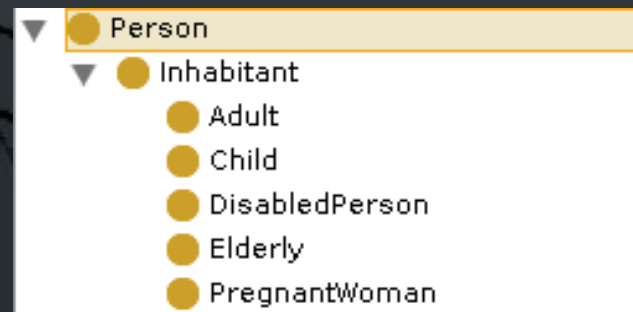
Other parameters ?

- Note that other parameters (time, surroundings of inhabitant, ...) can be included into the system in order to infer the inhabitant's health condition. But in this scenario, for simplicity, only Bpm and type of the action are the concepts used while detect user's health condition.



From Ontology perspective (1)

- Person class has different types of inhabitants. In a system with complete functionality, operation of inferring health condition must be done considering what type of inhabitant the person is.

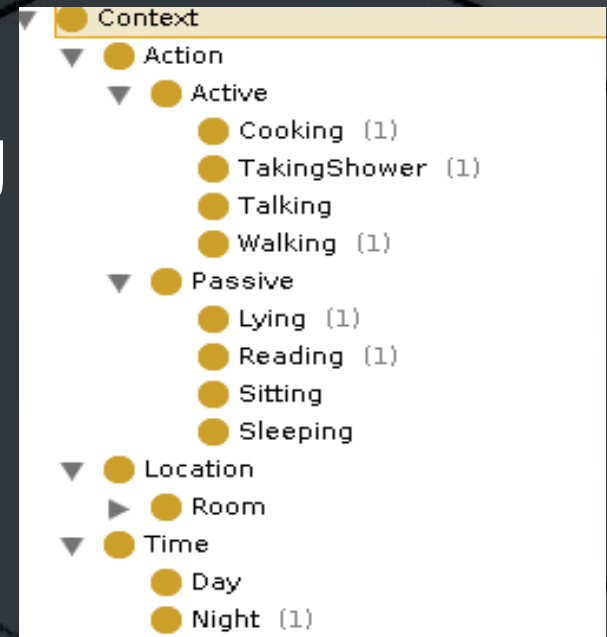


Note : Only the subclass “adult” is taken into consideration for simplicity of the scenario.

Example : The way a child behaves in the house is completely different with the way an adult does. That's why the action the child takes, the heart beat s/has would mean different thing while it would mean something else for an adult person.

From Ontology perspective (2)

- In context class, we have three subclasses;
 - Action → Active
 - * Taking a shower, walking
 - Action → Passive
 - * Sitting, sleeping, ...
 - Location → Room
 - * Kitchen, bedroom, ...
 - Time → Day, Night



Based on the user's context, the health condition is inferred. An example will be given later.

From Ontology perspective (3)

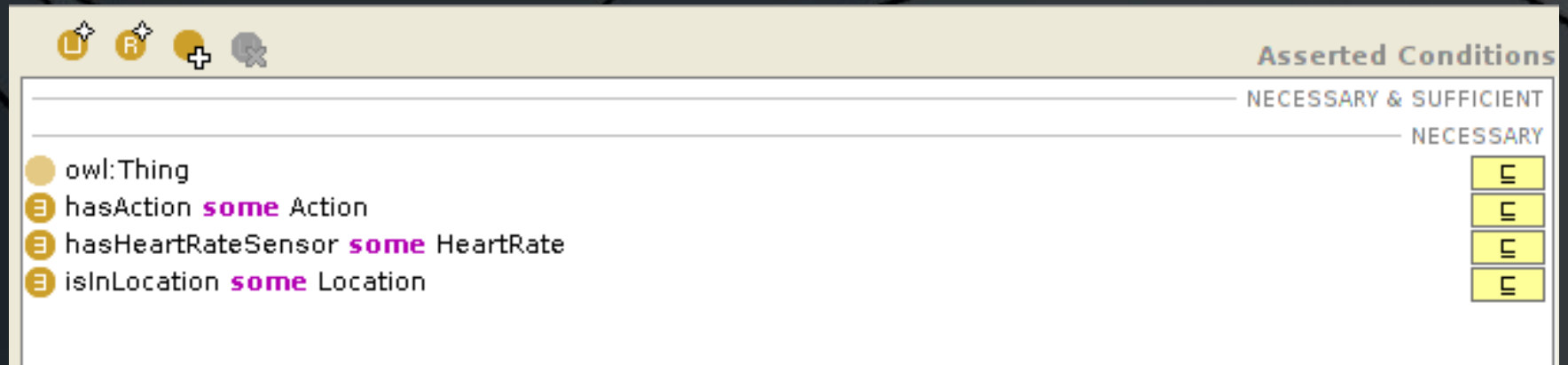
The screenshot displays an ontology editor interface with several property panels:

- hasName**: Instance: Arne
- hasHeartRateOf**: Instance: 85 (int). This instance is circled in green and labeled "Sensor Data".
- hasAction**: Instance: TakingShower_69
- isInLocation**: Instance: Bathroom_68
- durationPreference**: Instance: OneHourForShower
- hasHealthCondition**: Instance: LowRiskState_3 (Inferred). This instance is circled in green and labeled "Inferred". Below it, a tree view shows the hierarchy: HealthState (selected) -> AbnormalState -> LowRiskState (1).

* This inferring operation is done by the help of written SWRL rules.

Internal Rules - Restrictions

Example: The individuals that are to be created from Person class should have at least one action, heartRate and a location



The screenshot shows a software interface with a toolbar at the top containing icons for undo, redo, add, and delete. The main area is titled "Asserted Conditions" and contains a list of conditions. The conditions are:

- owl:Thing
- hasAction **some** Action
- hasHeartRateSensor **some** HeartRate
- isInLocation **some** Location

On the right side of the interface, there are four yellow buttons, each containing a subset symbol (⊆). The top two buttons are under the heading "NECESSARY & SUFFICIENT" and the bottom two are under "NECESSARY".

Note: If there is any individuals that does not meet these conditions, then inconsistency occurs.

External Rules - SWRL rules (examples)

1) The rule below defines a LowRisk state range for the persons who have a passive action and a heart rate between 81 – 120.

SWRL Rule

```
Person(?x) ∧ LowRiskState(?n) ∧ Passive(?y) ∧ hasAction(?x, ?y) ∧ hasHeartRateOf(?x, ?z) ∧ |swrlb:greaterThanOrEqual(?z, 81) ∧ swrlb:lessThanOrEqual(?z, 120) → hasHealthCondition(?x, ?n)
```

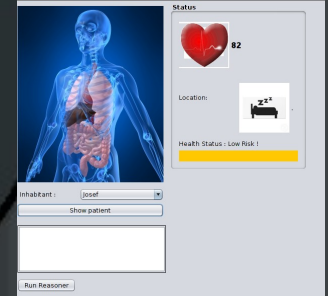
2) The rule below defines a HighRisk state range for the persons who have an active action and a heart rate greater than or equal to 131 Bpm.

SWRL Rule

```
Person(?x) ∧ HighRiskState(?n) ∧ Active(?y) ∧ hasAction(?x, ?y) ∧ hasHeartRateOf(?x, ?z) ∧ |swrlb:greaterThanOrEqual(?z, 131) → hasHealthCondition(?x, ?n)
```

* There are also other rules defined in the ontology for different ranges of health states.

Implementation



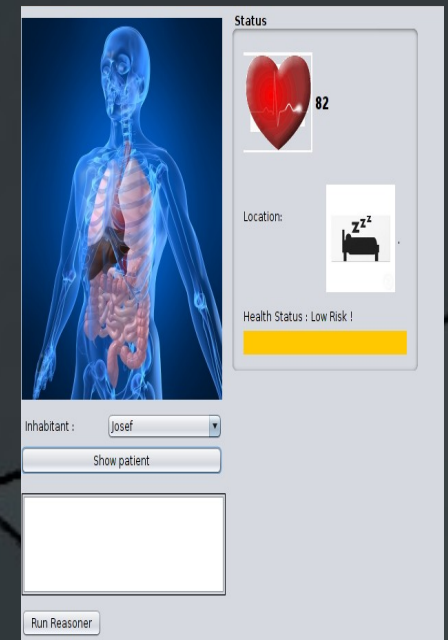
- Protege API is used in order to read, edit and write changes back into the ontology.
- CreateJenaOWLModelFromURI function is used in order to load ontology.
- There are some SWRL and SQRL queries written in API and used for several operations such as to find a person and show its related data and so on. To run these queries, query engine is used.
SWRLRuleEngine queryEngine =
SWRLRuleEngineFactory.create(owlM);

Inferring

- To run all of the defined rules, `inferAllRules` function is used .
- This function basically uses query engine.

```
queryEngine.importSWRLRulesAndOWLKnowledge();
```

- `queryEngine.run();`
- `queryEngine.infer();`
- `queryEngine.writeInferredKnowledge2OWL();`
- `writeToOntology();`
-



Implementation (1)

- A dummy heart beat sensor is created to generate random BPMs in every 3 seconds to mimic a real time application.
- Based on context of the user (only activity context in this application) and generated BPM value, user's health condition is updated each time heart beat is generated and shown in GUI.

Implementation (2)

Main functions:

- UpdateHealthStatusInOntology
 - load ontology, and selected person's data and show it .
- GetAskedPersonInfo
 - Access previously loaded owl and reach requested person's data (location, activity)
 - Each time when BPM changes, write it back to ontology with newly inferred health condition.

Future work (1)

- As stated before, only activity type (active and passive) and heartBeat values are taken into account when writing the rules for inferring health status of inhabitant. However more parameters (time, surroundings of inhabitant, preference of inhabitant) shall be added in order to cover different situations.
- And also only adult subclass of person class is considered in this application. However different types of scenarios could be considered based on user type (pregnant women, child, and so on).

Future work (2)

- Different types of sensor could be included such as temperature sensor, blood sugar sensor or some built-in sensors of smartphone. In this application, only heart beat sensor is simulated to keep application simple.

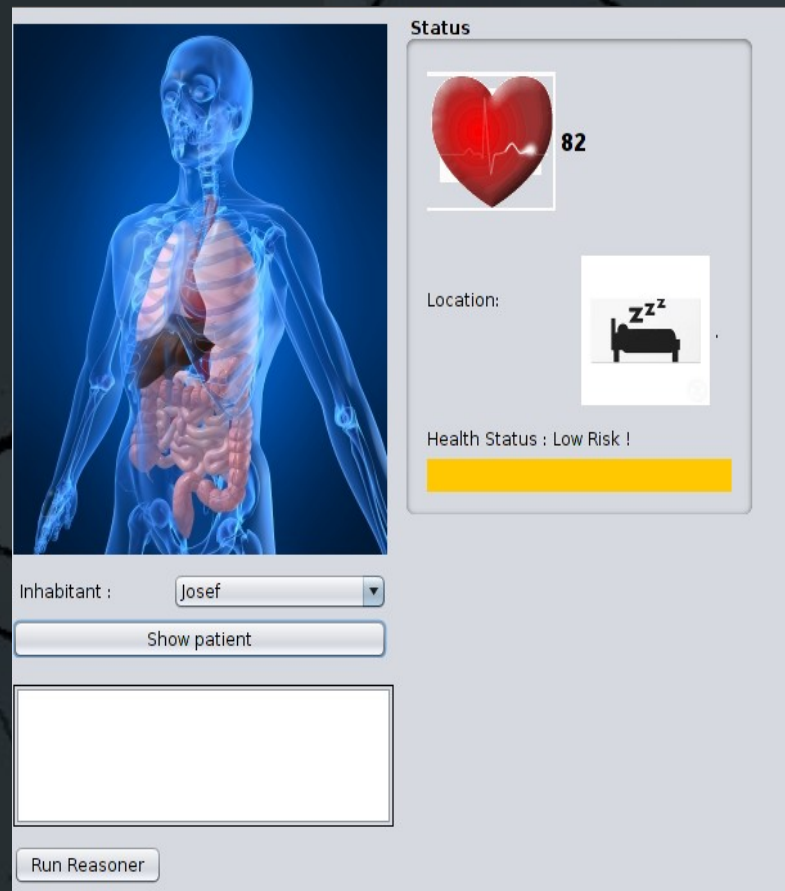
Difficulties met during the assignment

- Broken links : While getting benefit from Protege website, I have experienced some broken links that did not let me go into particulars.
- Pellet reasoner in Protege Editor: One thing bothered me while using Protege editor is that when I create a restriction for a class, and create an individual that disobeys this restriction, some unrelated inconsistencies occurred. I believe that this may cause some difficulties for owl developers to find out what the real inconsistencies are.

Some extra ideas

- The question is whether we will have some problems while writing the changes back into ontology in condition that we have a big ontology/ontologies. In my scenario, each time heart beat changes, system writes BPM back to ontology, checks what action s/he takes, and finds out the health condition and writes it into ontology (every 3 seconds) .
- I call my system real time considering the interaction between ontology and Protege API every 3 seconds . However, the definition of a real time application varies based on the system definition. For some applications this duration might be longer such as days, for some it could be seconds.
- Basically it is important to know what information needs to be distributed over the network and what inferred knowledge needs to be written right away. As an example from my health condition scenario, writing heart beat back into ontology every 3 seconds might not be necessary which leads us to change values on the air and show it just in GUI without writing into ontology. But, what happens if this kind of information needs to be shared by other users ?
- The point is that it is important to decide whether it is vital to write data back into owl or not . It completely depends on the aim of the application.

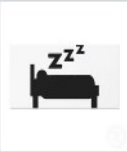
Thank you



The screenshot displays a medical application interface. On the left, there is a 3D anatomical model of a human torso with internal organs highlighted in red and blue. To the right of the model, the text 'Inhabitant : Josef' is shown next to a dropdown arrow. Below this is a 'Show patient' button. A large empty white rectangular area is positioned below the button. At the bottom left of the interface is a 'Run Reasoner' button.

Status

82

Location: 

Health Status : Low Risk !

Run Reasoner

Application Screenshot