Table of Contents

[1 Small Office Scenario – A Use Case: 1](#_Toc51854019)

[1.1 Instantiation of a Building Model: 1](#_Toc51854020)

[1.1.1 Route graph overlaid with building floor plan: 4](#_Toc51854021)

[1.1.2 Route Graph Generation: 6](#_Toc51854023)

[1.2 Instantiation of a User Model: 10](#_Toc51854024)

[1.3 Instantiation of a Context Model: 11](#_Toc51854025)

[2 Hazard Detection Context: A Task-based Evaluation Approach: 13](#_Toc51854026)

[2.1 Group-Based Activity: 16](#_Toc51854027)

[2.1.1 Family: 17](#_Toc51854028)

[2.1.2 Unfamiliar Visitor and a Leader: 22](#_Toc51854029)

[2.2 Autonomous Activity: 26](#_Toc51854030)

[2.2.1 Blind Person: 26](#_Toc51854031)

[2.2.2 Mobility-Impaired Person: 27](#_Toc51854032)

[2.2.3 Deaf Person: 29](#_Toc51854033)

# Small Office Scenario – A Use Case:

Here we describe a simple scenario in which SBEO is used as an example to define the semantics for a smart building evacuation system. Following are the details of the instantiation of building model, user model and context model with respect to a specific scenario.

## Instantiation of a Building Model:

Let us suppose a two-storey office space as shown in Figure 3, located in the big building arcade. It is mentioned using as an instance of **seas:Building** by using a **sbeo:partOf** property. Following is a fragment of these details of the layout of the building in the form of triples.

:OfficeBuilding1 rdf:type seas:OfficeBuilding ;

sbeo:partOf :ShoppingArcade .

:ShoppingArcade rdf:type sbeo:ShoppingMallBuilding .

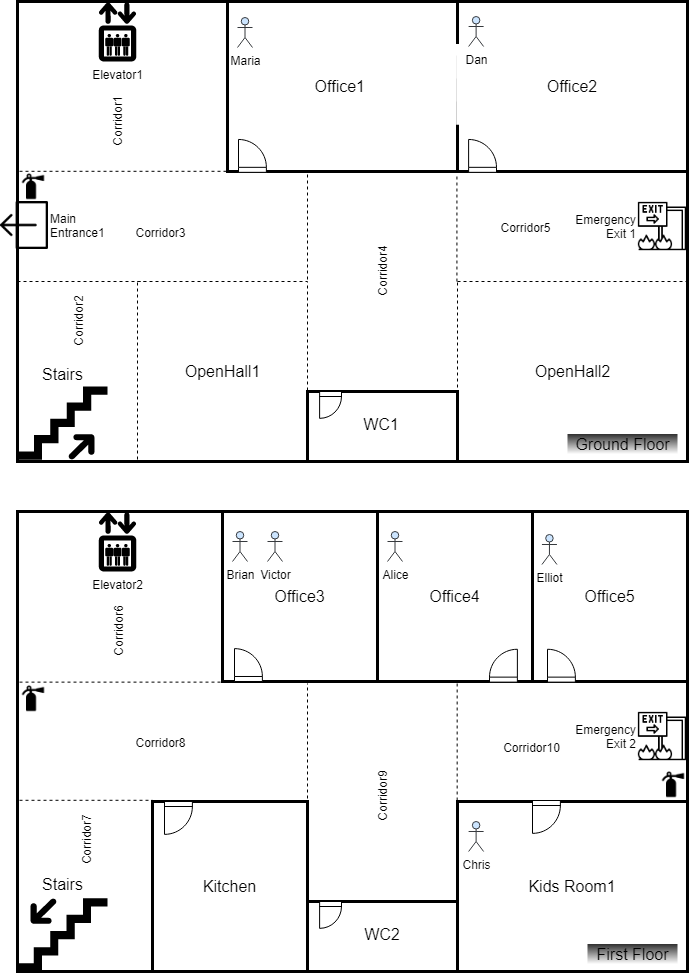
There are two offices, and one restroom located on the ground floor, and there are four offices, a kitchen and ****a kid’s room located on the first floor. There are elevator and stairs that are connected to both of the floors. This information is stored by creating the instances of **seas:BuildingSpace** and its sub-classes whereas the **sbeo:connectedTo**, **sbeo:locatedIn**, **sbeo:accommodatingCapacity**, **sbeo:hasLength**, and **sbeo:hasWidth** properties can also be used to describe the details of each space. Following is the list of some of the triples to describe the information about layout of the building.

Figure 3 BUILDING FLOOR PLAN FOR A SMALL OFFICE SPACE

Figure Building Floor Plan For a Small Office Space

:GroundFloor rdf:type seas:BuildingStorey ;

sbeo:locatedIn :OfficeBuilding1 .

:Office1 rdf:type seas:Office ;

sbeo:locatedIn :GroundFloor .

:Office2 rdf:type seas:Office ;

sbeo:locatedIn :GroundFloor .

:WC1 rdf:type seas:Bathroom ;

sbeo:locatedIn :GroundFloor .

:FirstFloor rdf:type seas:BuildingStorey ;

sbeo:locatedIn :OfficeBuilding1 .

:Office3 rdf:type seas:Office ;

sbeo:locatedIn :FirstFloor .

:Office4 rdf:type seas:Office ;

sbeo:locatedIn :FirstFloor .

:Office5 rdf:type seas:Office ;

sbeo:locatedIn :FirstFloor .

:WC2 rdf:type seas:Bathroom ;

sbeo:locatedIn :FirstFloor .

:Kitchen rdf:type seas:Kitchen ;

sbeo:locatedIn :FirstFloor .

:KidsRoom rdf:type sbeo:KidsArea ,

seas:Room ;

sbeo:locatedIn :FirstFloor .

:PV1 rdf:type seas:Elevator ;

sbeo:locatedIn :FirstFloor ,

:GroundFloor .

:PV2 rdf:type seas:Stairs ;

sbeo:locatedIn :FirstFloor ,

:GroundFloor .

The ground floor is connected to the main exit of the office space to the building. The office space also has a connection with the emergency exit of the main building, which is based on ramps. It means both the ground and first floor are connected to the emergency exit. This information is described by creating instances using **sbeo:Exit** and **sbeo:Door**. Following is a fragment of the information about the exits and entrances in the form of triples.

:FE1 rdf:type owl:NamedIndividual ,

sbeo:BuildingEntrance ,

sbeo:BuildingExit ,

seas:Door ,

sbeo:FloorExit ;

sbeo:locatedIn :GroundFloor .

:FE2 rdf:type owl:NamedIndividual ,

sbeo:FloorExit ,

sbeo:EmergencyExit ,

seas:Door ;

sbeo:locatedIn :GroundFloor .

:FE3 rdf:type owl:NamedIndividual ,

sbeo:FloorExit ,

sbeo:EmergencyExit ,

seas:Door ;

sbeo:locatedIn :FirstFloor .

The office space is considered as a *smart space* by the fact that it is equipped with the smoke, temperature and location sensors. It is mentioned using a type of a specific obersavle property. For example, **:Temperature**, **:Smoke** and **:PersonLocation** are the instances of **sosa:ObservableProperty** to sense the temperature, smoke and location of people respectively. Following is a fragment of the information about these sensors in the form of triples.

:TemperatureSensor42 rdf:type sosa:Sensor ;

sbeo:installedIn :Office2 ;

sosa:observes :Temperature .

:SmokeSensor41 rdf:type sosa:Sensor ;

sbeo:installedIn :Office1 ;

sosa:observes :Smoke .

:LocationSensor57 rdf:type sosa:Sensor ;

sbeo:installedIn :Kitchen ;

sosa:observes :PersonLocation .

:Temperature rdf:type sosa:ObservableProperty .

:Smoke rdf:type sosa:ObservableProperty .

:PersonLocation rdf:type sosa:ObservableProperty .

### Route graph overlaid with building floor plan:

Figure 4 shows a route graph that is overlaid with the building floor plan. This graph is generated by mapping the details of the building geometry with **sbeo:RouteElement** class in which edges and nodes are represented by **sbeo:Passage** and **sbeo:RoutePoint** respectively.

We can clearly see that the traversable areas of the building are represented as corridors, halls, or their segments that are expressed as edges, whereas the nodes are expressed as junctions, point of interests, end points, exits (rooms, floor and open area), and vertical passages. The reader may refer to the previous sections for the definition of each concept. Following is a fragment of the route elements transformed from the layout of the building.

:J1 rdf:type sbeo:Junction ;

sbeo:connectedTo :CorridorSegment1 ,

:CorridorSegment2 ,

:CorridorSegment3 ,

:CorridorSegment5 ;

sbeo:leadsTo :FE1 ,

:J2 ,

:PV1 ,

:PV2 ,

:PointOfInterest1.

:EP1 rdf:type sbeo:EndPoint ;

sbeo:connectedTo :CorridorSegment14 ;

sbeo:leadsTo :J3 .

:CorridorSegment14 rdf:type sbeo:CorridorSegment.

:Corridor1 rdf:type seas:Corridor ;

sbeo:connectedTo :Corridor3 ,

:PV1 ,

:J1 ,

:FE1;

sbeo:locatedIn :GroundFloor .

:PV1 rdf:type owl:NamedIndividual ,

seas:Elevator ;

sbeo:leadsTo :J1 ,

:J3 ;

sbeo:locatedIn :FirstFloor ,

:GroundFloor .

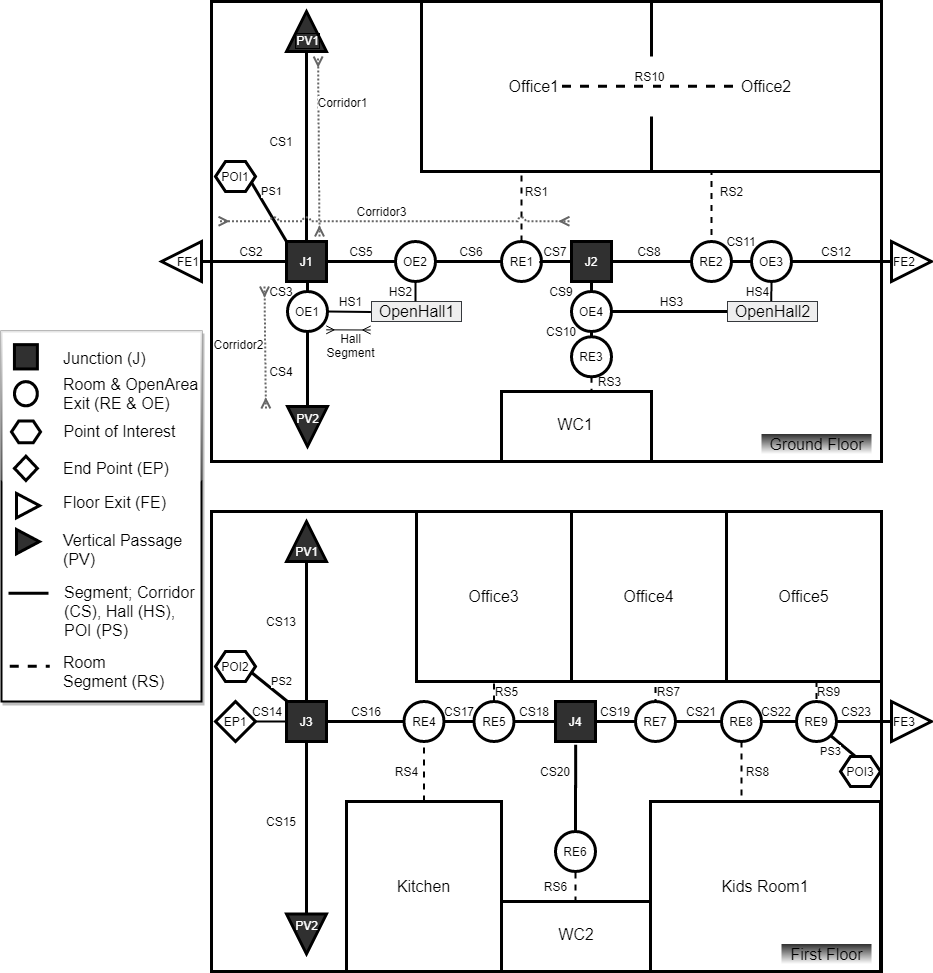


Figure ROUTE GRAPH OVERLAID WITH BUILDING FLOOR PLAN

### 

### Route Graph Generation:

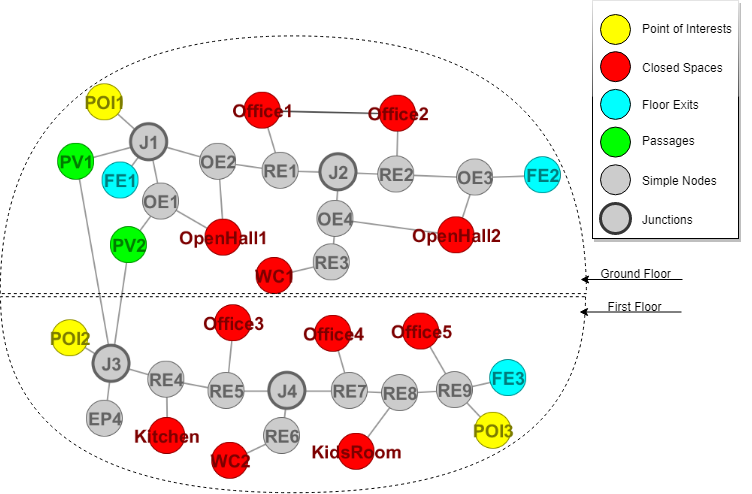
With the help of a route graph overlaid with building floor plan, a node-edge based undirected graph can be drawn. It is shown in Figure 5. This graph is created by expressing all the traversable areas such as corridors, vertical passages, roads, footpaths, etc., as edges whereas all the closed spaces, point of interests, junctions, exits, etc., as nodes. Note that, although the vertical passages (PV) are traversable, they are considered as nodes in our proposed system. It is not an issue because we can put an extra edge to mention their cost in the graph. But for sake of simplicity, we assume that the cost of these vertical passages is already added in the cost of the edges connected with them, e.g. the edge between PV1 and J1, and PV1 and J3. Similarly, for vertical passage PV2, the cost is either added to the edge between PV2 and J3, and PV2 and OE1.

Figure 5 AN UNDIRECTED GRAPH REPRESENTATION OF A BUILDING FLOOR PLAN

Later, we instantiate ten routes from the graph shown in Figure 5. Seven of them are exit routes, therefore they are the instances of **sbeo:ExitRoute** class whereas the other three routes are just the connection of different spaces that are used for transversal purposes within the building. These are shown in Figure 6. We have also explained in the figure that how each space is mentioned to form a route using ordered lists. For example, Route 1 is created using five slots and each slot has its own index number, item to mention the space, and the information about next or previous slot. Route 1 is given below in the form of triples.

:Route1 rdf:type olo:OrderedList ,

sbeo:ExitRoute ;

olo:slot :Slot1 ,

:Slot2 ,

:Slot3 ,

:Slot4 ,

:Slot5 ;

sbeo:length "5"^^xsd:nonNegativeInteger .

:Slot1 rdf:type olo:Slot ;

olo:item :J4 ;

olo:next :Slot2 ;

olo:ordered\_list :Route1 ;

olo:index "1"^^xsd:positiveInteger .

:Slot2 rdf:type olo:Slot ;

olo:item :RE7 ;

olo:next :Slot3 ;

olo:previous :Slot1;

olo:ordered\_list :Route1 ;

olo:index "2"^^xsd:positiveInteger .

:Slot3 rdf:type olo:Slot ;

olo:item :RE8 ;

olo:next :Slot4 ;

olo:previous :Slot2;

olo:ordered\_list :Route1 ;

olo:index "3"^^xsd:positiveInteger .

:Slot4 rdf:type olo:Slot ;

olo:item :RE9 ;

olo:next :Slot5 ;

olo:previous :Slot3;

olo:ordered\_list :Route1 ;

olo:index "4"^^xsd:positiveInteger .

:Slot5 rdf:type olo:Slot ;

olo:item :FE3 ;

olo:previous :Slot4;

olo:ordered\_list :Route1 ;

olo:index "5"^^xsd:positiveInteger .

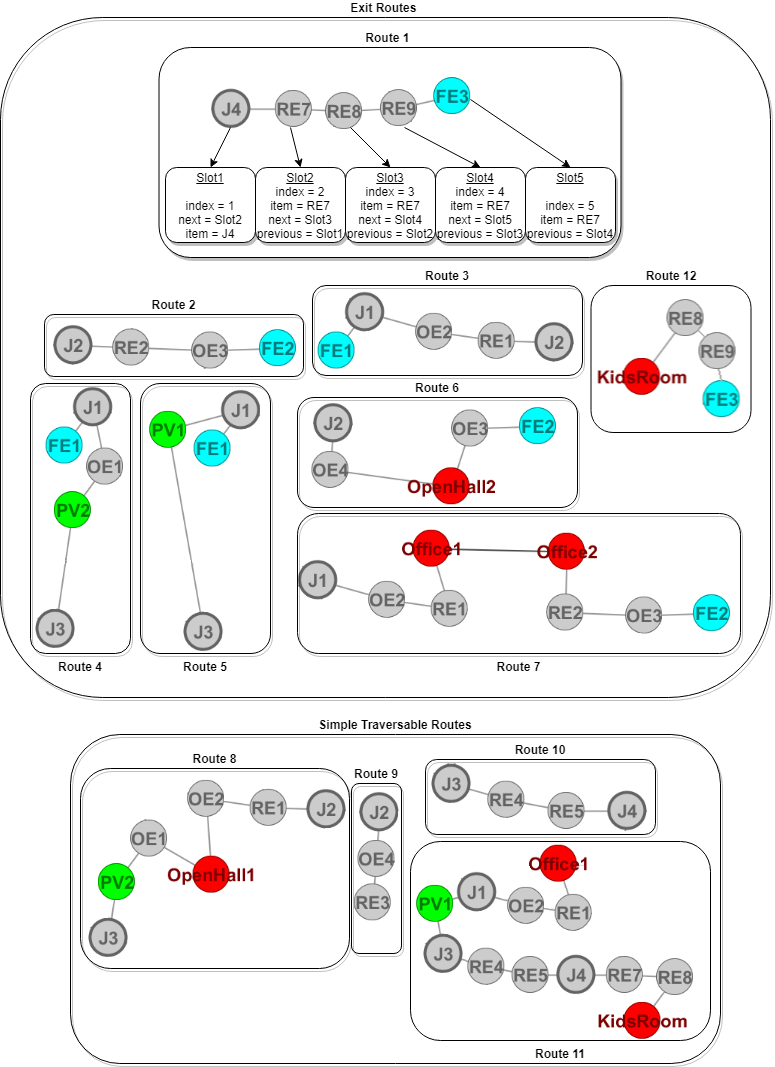


Figure 6 Various Types of Routes Derived From the Graph

## Instantiation of a User Model:

As it is a working day, therefore seven persons named, Alice (Uses motorized wheelchair), Brian, Chris (family ties with Maria, kid), Dan (Blind Person), Elliot (Deaf Person), Maria (family ties with Chris, Mother), are located in the office space. Six of them are there for work purposes, and one person (Chris) is a child who has an off-day from school today. His mother (Maria) brought him to her office and dropped him at Kids’ Room which is located on the first floor of the office space. Maria’s office is located on the ground floor adjacent to the office of Elliot. The rest of the persons have offices on the first floor. On the meantime, a person named Victor visits the office who has a meeting with Brian on the first floor. These details are mentioned using by creating the instances of **sbeo:Family**, **foaf:Person**, and **sbeo:ImpairedPerson** class. Following is a fragment of the above-mentioned information in the form of triples.

:Person1 rdf:type sbeo:MotorisedWheelchairPerson ;

foaf:firstName "Alice"^^xsd:string ;

foaf:age 35 .

:Person2 rdf:type foaf:Person ;

foaf:firstName "Brian"^^xsd:string ;

foaf:age 55 .

:Person3 rdf:type foaf:Person ;

foaf:firstName "Chris"^^xsd:string ;

foaf:age 12 .

:Person4 rdf:type sbeo:BlindPerson ;

foaf:firstName "Dan"^^xsd:string ;

foaf:age 28 .

:Person5 rdf:type sbeo:DeafPerson ;

foaf:firstName "Elliot"^^xsd:string ;

foaf:age 37 .

:Person6 rdf:type foaf:Person ;

foaf:firstName "Maria"^^xsd:string ;

foaf:age 34 .

:Person7 rdf:type foaf:Person ;

foaf:firstName "Victor"^^xsd:string ;

foaf:age 25 .

:Family1 rdf:type sbeo:Family ;

sbeo:hasMember :Person3 ,

:Person6 .

Everyone except Victor and Chris, is familiar with the geometry of the office space which is expressed using a **sbeo:familiarWith** property whose object is **sbeo:OfficeBuilding**. Following is the list of triples associated with people who are familiar with the building.

:Person1 sbeo:familiarWith :OfficeBuilding1 .

:Person2 sbeo:familiarWith :OfficeBuilding1 .

:Person4 sbeo:familiarWith :OfficeBuilding1 .

:Person5 sbeo:familiarWith :OfficeBuilding1 .

:Person6 sbeo:familiarWith :OfficeBuilding1 .

:OfficeBuilding1 rdf:type seas:OfficeBuilding .

Each person is associated with a **sbeo:meansOfNotification** property. For example, Alice, Brian, Elliot and Maria are informed by their phones whereas Chris is informed by a screen because the Kid’s Room and Kitchen is equipped with a Digital Screens and speakers which are the instances of **sbeo:AlertingDevice** and **sbeo:DisplayScreen** concepts. Following is the list of triples which are used to express the means of notification of each person.

:Person1 sbeo:meansOfNotification :Phone1 .

:Person2 sbeo:meansOfNotification :PDA1 .

:Person3 sbeo:meansOfNotification :Screen1 .

:Person4 sbeo:meansOfNotification :SpecialPurposeDevice1 .

:Person5 sbeo:meansOfNotification :Phone3 .

:Person6 sbeo:meansOfNotification :Phone4 .

:Phone1 rdf:type sbeo:SmartPhone .

:Phone3 rdf:type sbeo:SmartPhone .

:Phone4 rdf:type sbeo:SmartPhone .

:PDA1 rdf:type sbeo:HandheldDevice .

:Screen1 rdf:type sbeo:DisplayScreen ;

sbeo:installedIn :KidsRoom .

:SpecialPurposeDevice1 rdf:type sbeo:AlertingDeviceForImpairedPerson ,

sbeo:HandheldDevice .

## Instantiation of a Context Model:

The role of each person is mention using **sbeo:hasRole** property in which Alice, Dan and Elliot have Autonomous roles, Brian and Maria have a role of responsible persons, and Chris and Victor have the role of dependent persons by the fact that both of them are not familiar with the geometry of the building. Following is the list of triples used to express the role of each person.

:Person1 sbeo:hasRole sbeo:Autonomous .

:Person2 sbeo:hasRole sbeo:Responsible .

:Person3 sbeo:hasRole sbeo:Dependent .

:Person4 sbeo:hasRole sbeo:Autonomous .

:Person5 sbeo:hasRole sbeo:Autonomous .

:Person6 sbeo:hasRole sbeo:Responsible .

:Person7 sbeo:hasRole sbeo:Dependent .

Persons who have Autonomous and Responsible roles are equipped with the mobile or Personal Digital Assistant provided by their company that can be used during any emergency. This information about the device is described by instantiating the **sbeo:HandheldDevice** class, and the person who are using these devices are expressed with the help of **sbeo:uses** property. These handheld devices have software in which everyone is registered with his or her abilities and preferences. Following is the list of triples used to express which device is being used by whom.

:Person1 sbeo:uses :Phone1 .

:Person2 sbeo:uses :PDA1 .

:Person4 sbeo:uses :SpecialPurposeDevice1 .

:Person5 sbeo:uses :Phone3 .

:Person6 sbeo:uses :Phone4 .

:Phone1 rdf:type sbeo:SmartPhone .

:Phone3 rdf:type sbeo:SmartPhone .

:Phone4 rdf:type sbeo:SmartPhone .

:PDA1 rdf:type sbeo:HandheldDevice .

:SpecialPurposeDevice1 rdf:type sbeo:AlertingDeviceForImpairedPerson ,

sbeo:HandheldDevice .

These preferences are expressed on the basis of various properties such as **sbeo:navigationalType**, **sbeo:meansOfNotification**, and **sbeo:routePreference** etc.

For example, in case of **sbeo:navigationalType**, Alice, Dan, and Elliot have *sbeo:AssistedNavigation*, whereas Brian, Maria, Chris and Victor have *sbeo:CollaborativeNavigation*. Finally, Maria has *sbeo:MultiObjectiveNavigation* as well. With a help of an algorithm, these navigation types are inferred on the basis of their characteristics and preferences. Following is the list of triples used to express the navigational type of each person.

:Person1 sbeo:hasNavigationalType sbeo:AssistedNavigation .

:Person2 sbeo:hasNavigationalType sbeo:AssistedNavigation ,

sbeo:CollaborativeNavigation.

:Person3 sbeo:hasNavigationalType sbeo:CollaborativeNavigation .

:Person4 sbeo:hasNavigationalType sbeo:AssistedNavigation .

:Person6 sbeo:hasNavigationalType sbeo:AssistedNavigation,

sbeo:CollaborativeNavigation ,

sbeo:MultiObjectiveNavigation .

:Person7 sbeo:hasNavigationalType sbeo:CollaborativeNavigation .

# Hazard Detection Context: A Task-based Evaluation Approach:

Suddenly, the smoke and temperature sensors detect the increase in temperature and existence of smoke in the kitchen with the help of **sbeo:hasSmoke** and **sbeo:hasValue**, and **sbeo:atTime** properties. Following is the list of relevant triples. Note that the triples mentioned in the green boxes with a single-lined frame represent the static information which is added at the time of designing the system and acquisition of knowledge. On the other hand, the triples mentioned in the yellow boxes with a double-lined frame represent the dynamic information which is added into the ontology on real-time basis, depending on the scenario, event, incident, and so forth. Although it is not necessary to mention about this static information here, we have stated it just for the sake of clarity so that a potential reader can understand the context.

**Step 1: Relevant triples in the existing knowledge base.**

:TemperatureSensor57 rdf:type sosa:Sensor ;

sbeo:installedIn :Kitchen ;

sosa:observes :Temperature .

:Kitchen rdf:type seas:Kitchen .

:SmokeSensor57 rdf:type sosa:Sensor ;

sbeo:installedIn :Kitchen ;

sosa:observes :Smoke .

:Kitchen rdf:type seas:Kitchen .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:TemperatureSensor57 sbeo:hasValue “55”^^xsd:integer ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

:SmokeSensor57 sbeo:hasSmoke “True”^^xsd:boolean ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp.

The fire incidence is created as an instance of **sbeo:Fire** (which is a subclass of **sbeo:Incidence** concept). The starting time of this fire event is also stored with the help of **sbeo:startedAtTime** property. The momentary information about the existence of fire is updated using timestamps after fixed time intervals using **sbeo:atTime** property (but it is discussed here in detail due the scope of this document). Following is the list of relevant triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Fire1 rdf:type sbeo:Fire .

sbeo:locatedIn :Kitchen .

:Kitchen rdf:type seas:Kitchen .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Fire1 sbeo:startedAtTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

sbeo:atTime "2020-07-29T00:09:15Z"^^xsd:dateTimeStamp .

As soon as the fire is detected, everyone is notified about the fire event on the device of their choice or being recommended by default using **sbeo:meansOfNotification** property. We have already mentioned the triples that express the means of notification of each person.

Everyone has asked to evacuate the office space as soon as possible with the help of a message. This message has been sent to all notification devices using a **sbeo:dynamicDescription** property. Following is the list of triples to express the above-mentioned message on relevant devices.

**Step 1: Relevant triples in the existing knowledge base.**

:Phone1 rdf:type sbeo:SmartPhone .

:Phone3 rdf:type sbeo:SmartPhone .

:Phone4 rdf:type sbeo:SmartPhone .

:PDA1 rdf:type sbeo:HandheldDevice .

:Screen1 rdf:type sbeo:DisplayScreen ;

sbeo:installedIn :KidsRoom .

:SpecialPurposeDevice1 rdf:type sbeo:AlertingDeviceForImpairedPerson ,

sbeo:HandheldDevice .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Phone1 sbeo:dynamicDescription "Please, evacuate the building."^^xsd:string .

:Phone3 sbeo:dynamicDescription "Please, evacuate the building."^^xsd:string .

:Phone4 sbeo:dynamicDescription "Please, evacuate the building."^^xsd:string .

:PDA1 sbeo:dynamicDescription "Please, evacuate the building."^^xsd:string .

:Screen1 sbeo:dynamicDescription "Please, evacuate the building."^^xsd:string .

:SpecialPurposeDevice1 sbeo:dynamicDescription "Please, evacuate the

building."^^xsd:string .

Due to the fire event, the system has blocked the access to the kitchen by updating its status from *sbeo:Available* to *sbeo:UnAvailable* using **sbeo:hasAvailabilityStatus** property as shown below. Following are the triples to express this property. The triple mentioned in the blue box with a dash-lined frame represents that it is removed from the system, whereas a new triple is added which is represented in a yellow box. By default, all the spaces have **sbeo:hasAvailabilityStatus**, *sbeo:Available,* therefore, the triple to express the availability of a particular space is written in green box. For the sake of understanding, we divide the process into steps.

**Step 1: Relevant triples in the existing knowledge base.**

:Kitchen rdf:type seas:Kitchen ;

sbeo:hasAvailabilityStatus sbeo:Available .

**Step 2: Triples those are to be removed from the knowledge base .**

:Kitchen sbeo:hasAvailabilityStatus sbeo:Available .

**Step 3: Relevant triples those are remained in the knowledge base (only applies to this example).**

:Kitchen rdf:type seas:Kitchen .

**Step 4: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Kitchen sbeo:hasAvailabilityStatus sbeo:UnAvailable .

With the help of an algorithm, the system has calculated the type of possible evacuation routes for each person based on their characteristics and preferences. These routes are calculated with the help of generating a graph based on nodes and edges as mentioned in the previous section. For example, **sbeo:SimplestPath** is assigned as a **sbeo:routePreference** of blind and mobility impaired persons (also known as motor impaired persons). On the other hand, **sbeo:ShorestPath** for all other types of persons. Following is a list of triples expressing the route preference of each person with respect to one’s type.

**Step 1: Relevant triples in the existing knowledge base.**

:Person1 rdf:type sbeo:MotorisedWheelchairPerson .

:Person2 rdf:type foaf:Person .

:Person3 rdf:type foaf:Person .

:Person4 rdf:type sbeo:BlindPerson .

:Person5 rdf:type sbeo:DeafPerson .

:Person6 rdf:type foaf:Person .

:Person7 rdf:type foaf:Person .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person1 sbeo:routePreference sbeo:SimplestPath .

:Person2 sbeo:routePreference sbeo:ShortestPath .

:Person3 sbeo:routePreference sbeo:ShortestPath .

:Person4 sbeo:routePreference sbeo:SimplestPath .

:Person5 sbeo:routePreference sbeo:ShortestPath .

:Person6 sbeo:routePreference sbeo:ShortestPath .

:Person7 sbeo:routePreference sbeo:ShortestPath .

Everybody is working in their offices, while Chris is playing in the Kid’s Area. The information about the people located in these spaces with specific timestamps is stored in the server with the help of **sbeo:locatedIn** and **sbeo:atTime** properties. Following is the fragment of triples which is used to express the above-mentioned information.

:Person1LocationEvent rdf:type sbeo:LocationSensorEvent ;

sbeo:forPerson :Person1 ;

sbeo:locatedIn :Office4 ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

:Person2LocationEvent rdf:type sbeo:LocationSensorEvent ;

sbeo:forPerson :Person2 ;

sbeo:locatedIn :Office3 ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

:Person3LocationEvent rdf:type sbeo:LocationSensorEvent ;

sbeo:forPerson :Person3 ;

sbeo:locatedIn :KidsRoom ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

:Person4LocationEvent rdf:type sbeo:LocationSensorEvent ;

sbeo:forPerson :Person4 ;

sbeo:locatedIn :Office2 ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

:Person5LocationEvent rdf:type sbeo:LocationSensorEvent ;

sbeo:forPerson :Person5 ;

sbeo:locatedIn :Office5 ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

:Person6LocationEvent rdf:type sbeo:LocationSensorEvent ;

sbeo:forPerson :Person6 ;

sbeo:locatedIn :Office1 ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

:Person7LocationEvent rdf:type sbeo:LocationSensorEvent ;

sbeo:forPerson :Person7 ;

sbeo:locatedIn :Office3 ;

sbeo:atTime "2020-07-29T00:09:00Z"^^xsd:dateTimeStamp .

The people who are familiar with the layout of the building are also found using **sbeo:familiarWith** properties. The list of triples to express the familiarity is already mentioned before.

For the sake of understanding, here we divide the use-case with respect to the roles and types of each person. It is obvious that either a person will belong to group in which one’s role will be either dependent or responsible which performing any specific activity, or the person will be autonomous. Following is the classification of roles and types of people with respect to the current use-case of emergency evacuation.

## Group-Based Activity:

### Family:

Later, the system checks who needs to be stayed at their places to avoid panic, such as children, etc. Following is a fragment of triples to express the location of children located in the same building where the fire incident occurred.

**Step 1: Relevant triples in the existing knowledge base.**

:Person3 rdf:type foaf:Person;

foaf:age “12”xsd:integer .

:KidsRoom rdf:type sbeo:KidsArea ;

sbeo:locatedIn :FirstFloor .

:Kitchen rdf:type seas:Kitchen ;

Sbeo:locatedIn :FirstFloor .

:FirstFloor sbeo:locatedIn :OfficeBuilding1 .

:OfficeBuilding1 rdf:type seas:OfficeBuilding .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person3 sbeo:locatedIn :KidsRoom .

:Fire1 rdf:type sbeo:Fire ;

sbeo:locatedIn :Kitchen .

The message is displayed on the screen to Chris that his mother; Maria, is coming to pick him up and he has to keep waiting for her. It is done with the help of the **sbeo:dynamicDescription** property whose domain is **sbeo:Device** and range is **xsd:string**. Following is an example of the triple.

**Step 1: Relevant triples in the existing knowledge base.**

:Screen1 rdf:type sbeo:DisplayScreen ;

sbeo:locatedIn :KidsRoom .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Screen1 sbeo:dynamicDescription “Chris, your mother is coming to pick you up. Please, stay

at your place.”^^xsd:string .

Maria has become a **sbeo:Responsible** person (and responsible to Chris) of this group whereas Chris has become a **sbeo:Dependent** person (and dependent on Maria). Following are the triples to express the above-mentioned information.

**Step 1: Relevant triples in the existing knowledge base.**

:Family1 rdf:type sbeo:Family .

sbeo:hasMember :Person3 ,

:Person6 .

:Person6 rdf:type foaf:Person ;

foaf:firstName “Maria”^^xsd:string ;

foaf:age 34.

:Person3 rdf:type foaf:Person ;

foaf:firstName “Chris”^^xsd:string ;

foaf:age 12.

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person6 sbeo:hasRole sbeo:Responsible ;

sbeo:responsibleTo :Person3 .

:Person3 sbeo:hasRole sbeo:Dependent ;

sbeo:dependentOn :Person6 .

As Maria is an able-bodied person and an instance of **foaf:Person**, therefore the notification preference of Maria is **sbeo:TextualDescription**. Following is an example of above-mentioned information in the form of triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Person6 rdf:type foaf:Person .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person6 sbeo:notificationPreference :Email .

:Email rdf:type sbeo:TextualDescription .

We have mentioned previously that Maria has **sbeo:navigationalType** *sbeo:MultiObjectiveNavigiation, sbeo:AssistedNavigation and sbeo:CollaborativeNavigation.* The **sbeo:ShortestRoute** (in terms of time); an elevator one, is calculated from the location of Maria to the location of Chris and assigned to Maria with the help of **sbeo:assignedRoute**. Following is a fragment of above-mentioned information in the form of triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Person6 rdf:type foaf:Person ;

sbeo:routePreference sbeo:ShortestPath ;

sbeo:locatedIn :Office1 .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person6 sbeo:assignedRoute :Route11 .

Once Maria starts to follow the provided route, her **sbeo:activityStatus** becomes *sbeo:Evacuating* as well as *sbeo:PickingUpDependents*. Following is a fragment of information in the form of the triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Person6 rdf:type foaf:Person .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person6 sbeo:hasActivityStatus sbeo:Evacuating ,

sbeo:PickingUpDependents .

As soon as Maria picks Chris up, both of them become an instance of **sbeo:EmergencyEvacuationGroup** class by creating *:Family1* group as an instance of it. Because they are already assigned as a group. This information is given below in the form of triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Family1 rdf:type sbeo:Family ;

sbeo:hasMember :Person3 ,

:Person6 .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Family1 rdf:type sbeo:EmergencyEvacuationGroup .

Also, when Maria reaches Chris’s room. Her activity status of *sbeo:PickingUpDependents* is removed and only *sbeo:Evacuating* remains. Also, the activity status of Chris as well as the whole group become *sbeo:Evacuating* too. Following is a fragment of information in the form of the triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Person6 rdf:type foaf:Person ;

sbeo:hasActivityStatus sbeo:Evacuating ,

sbeo:PickingUpDependents .

:Person3 rdf:type foaf:Person .

:Family1 rdf:type sbeo:Family ,

sbeo:EmergencyEvacuationGroup .

**Step 2: Triples those are to be removed from the knowledge base.**

:Person6 sbeo:hasActivityStatus sbeo:PickingUpDependents .

**Step 3: Relevant triples those are remained in the knowledge base (only applies to this example).**

:Person6 rdf:type foaf:Person ;

sbeo:hasActivityStatus sbeo:Evacuating .

:Person3 rdf:type foaf:Person .

:Family1 rdf:type sbeo:Family ,

sbeo:EmergencyEvacuationGroup .

**Step 4: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person3 sbeo:hasActivityStatus sbeo:Evacuating .

:Family1 sbeo:hasActivityStatus sbeo:Evacuating .

In the meantime, the smoke has been spread across the elevator and stairs, and as a result, they have become unavailable. This information is given below in the form of triples.

**Step 1: Relevant triples in the existing knowledge base.**

:SmokeSensor76 rdf:type sosa:Sensor ;

sbeo:installedIn :PV1 ;

sosa:observes :Smoke .

:SmokeSensor77 rdf:type sosa:Sensor;

sbeo:installedIn :PV2 ;

sosa:observes : Smoke .

:PV2 rdf:type seas:Stairs ;

sbeo:hasAvailabilityStatus sbeo:Available .

:PV1 rdf:type seas:Elevator ;

sbeo:hasAvailabilityStatus sbeo:Available .

**Step 2: Triples those are to be removed from the knowledge base.**

:PV2 sbeo:hasAvailabilityStatus sbeo:Available .

:PV1 sbeo:hasAvailabilityStatus sbeo:Available .

**Step 3: Relevant triples those are remained in the knowledge base (only applies to this example).**

:SmokeSensor76 sbeo:hasSmoke “True”^^xsd:boolean ;

sbeo:atTime "2020-07-29T00:09:17Z"^^xsd:dateTimeStamp.

:SmokeSensor77 sbeo:hasSmoke “True”^^xsd:boolean ;

sbeo:atTime "2020-07-29T00:09:17Z"^^xsd:dateTimeStamp.

:PV2 sbeo:hasAvailabilityStatus sbeo:unAvailable .

:PV1 sbeo:hasAvailabilityStatus sbeo:unAvailable .

As a result, that area is no longer accessible for this group (*:Family1*). This is done using **sbeo:excludedFor** property whose domain is **sbeo:Space** and range is **sbeo:SocialUnit**. Even though the stairs can be used by them, but it was stored in the system (**sbeo:excludedFor**) that the area which has smoke can be dangerous for the group consists of children. Therefore, it was automatically filtered out while computing the exit routes for them. Following are the triples to express the above-mentioned information.

**Step 1: Relevant triples in the existing knowledge base.**

:PV2 rdf:type seas:Stairs .

:PV1 rdf:type seas:Elevator .

:Family1 rdf:type sbeo:Family ,

sbeo:EmergencyEvacuationGroup .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:PV2 sbeo:excludedFor :Family1 .

:PV1 sbeo:excludedFor :Family1 .

Consequently, *:Route12* is provided to Maria and Chris that leads to an emergency exit. The list of triples to express the above-mentioned information is given below.

**Step 1: Relevant triples in the existing knowledge base.**

:Family1 rdf:type sbeo:Family ,

sbeo:EmergencyEvacuationGroup ;

sbeo:routePreference sbeo:ShortestRoute .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Family1 sbeo:assignedRoute :Route12 .

as soon as Maria, and Chris leave the office space, the **sbeo:activityStatus** of both of them turns to be ‘*sbeo:Evacuated*’ from ‘*sbeo:Evacutating*’.

**Step 1: Relevant triples in the existing knowledge base.**

:Person6 rdf:type foaf:Person ;

sbeo:hasActivityStatus sbeo:Evacuating .

:Person3 rdf:type foaf:Person ;

sbeo:hasActivityStatus sbeo:Evacuating .

:Family1 rdf:type sbeo:Family ,

sbeo:EmergencyEvacuationGroup ;

sbeo:hasActivityStatus sbeo:Evacuating .

**Step 2: Triples those are to be removed from the knowledge base.**

:Person6 sbeo:hasActivityStatus sbeo:Evacuating .

:Person3 sbeo:hasActivityStatus sbeo:Evacuating .

:Family1 sbeo:hasActivityStatus sbeo:Evacuating .

**Step 3: Relevant triples those are remained in the knowledge base (only applies to this example).**

:Person6 rdf:type foaf:Person .

:Person3 rdf:type foaf:Person .

:Family1 rdf:type sbeo:Family ,

sbeo:EmergencyEvacuationGroup .

**Step 4: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person6 sbeo:hasActivityStatus sbeo:Evacuated .

:Person3 sbeo:hasActivityStatus sbeo:Evacuated .

:Family1 sbeo:hasActivityStatus sbeo:Evacuated .

### Unfamiliar Visitor and a Leader:

Brian and Victor are located at Brian’s office, which is on the first floor of the office space. Following is the list of relevant triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Person2 rdf:type foaf:Person ;

foaf:firstname “Dan”^^xsd:string .

:Person7 rdf:type foaf:Person ;

foaf:firstname “Victor”^^xsd:string .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person2 sbeo:locatedIn :Office3 .

:Person7 sbeo:locatedIn :Office3 .

In addition, as both Brian and Victor are located at a same place, they have become the members of same group named *:EmergencyEvacuationGroup1* which is an instance of **sbeo:EmergencyEvacuationGroup**.This information is given below in the form of triples.

:EmergencyEvacuationGroup1 rdf:type sbeo:EmergencyEvacuationGroup ;

sbeo:hasMember :Person2 ,

:Person7 .

As Victor is not familiar with the geometry of the building, therefore his **sbeo:hasRole** is considered as ***sbeo:Dependent*** by the system based on the fact that he is located at a space where there is another person who is familiar with the layout of the building. The list of triples to express the above-mentioned information is given below.

**Step 1: Relevant triples in the existing knowledge base.**

:Person7 rdf:type foaf:Person ;

foaf:age “25”^^xsd:integer ;

sbeo:hasRole sbeo:Dependent .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person7 sbeo:hasRole sbeo:Dependent .

In this group, Brian has a role of *sbeo:Leader* and *sbeo:Responsible* whereas Victor has a role of *sbeo:Dependent* member. In other words, Victor is obliged to follow Brian’s instruction. Following are the relevant triples to express this information.

**Step 1: Relevant triples in the existing knowledge base.**

:Person2 rdf:type foaf:Person .

:Person7 rdf:type foaf:Person .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person2 sbeo:hasRole sbeo:Leader ,

sbeo:Responsible .

:Person7 sbeo:hasRole sbeo:Dependent .

Although Victor is young, Brian is in his late 50’s (**using foaf:age**). Therefore, both of them are provided by a route that is compromised on an elevator with the help of **sbeo:excludedFor** property in which **seas:Stairs** is used as domain and **foaf:Person** (Brian) is used as a range. Following is a fragment of information in the form of the triples.

**Step 1: Relevant triples in the existing knowledge base.**

:PV2 rdf:type seas:Stairs .

:EmergencyEvacuationGroup1 rdf:type sbeo:EmergencyEvacuationGroup .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:PV2 sbeo:excludedFor :EmergencyEvacuationGroup1 .

Brian **sbeo:uses** a Personal Digital Assistant as his his **sbeo:meansOfNotification**, which is represented as *:PDA1* (which is an instance of **sbeo:HandheldDevice**), where all the information is provided. Following are the triples to express a means of notification of him.

**Step 1: Relevant triples in the existing knowledge base.**

:Person2 sbeo:uses :PDA1 .

:PDA1 rdf:type sbeo:HandheldDevice .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person2 sbeo:meansOfNotification :PDA1 .

The **sbeo:notificationPreference** of Brian is both **sbeo:ImageDescription** and **sbeo:AudioDescription** by the fact that he is in his 60’s and it will be easier for him to visualize his route as well as hear the instructions related to it. Following are the triples to is an example of the triple.

**Step 1: Relevant triples in the existing knowledge base.**

:Person2 sbeo:uses :PDA1 .

:PDA1ImageDescription rdf:type sbeo:ImageDescription .

:PDA1AudioDescription rdf:type sbeo:AudioDescription .

**Step 2: Triples those are to be added in the knowledge base dynamically (only applied on this example).**

:Person2 sbeo:notificationPreference :PDA1ImageDescription ,

:PDA1AudioDescription .

Note that, in general, the usage of elevators is avoided during hazardous situations such as a fire. But in this scenario, the **sbeo:availabilityStatus** of *:PV1* (instance of **sbeo:Elevator**) is still *sbeo:Available* yet, by the fact that the smoke has not been detected in its surroundings. Therefore, Brian and Victor have been assigned a route in which elevator is included. Following is the list of relevant triples to express this information.

:SmokeSensor76 rdf:type sosa:Sensor ;

sbeo:installedIn :PV1 ;

sosa:observes :Smoke ;

sbeo:hasSmoke “False”^^xsd:boolean ;

sbeo:atTime "2020-07-29T00:09:16Z"^^xsd:dateTimeStamp.

:PV1 rdf:type seas:Elevator .

sbeo:hasAvailabilityStatus sbeo:Available .

The information about their **sbeo:Route** is also given to the **sbeo:HandheldDevice**, which is assigned using **sbeo:assignedRoute** property. As their route preference is a shortest path, therefore they have been assigned with *:Route17*. Following are the triples to express this information.

**Step 1: Relevant triples in the existing knowledge base.**

:EmergencyEvacuationGroup1 rdf:type sbeo:EmergencyEvacuationGroup .

:EmergencyEvacuationGroup1 sbeo:routePreference sbeo:ShortestPath ;

sbeo:assignedRoute :Route17 .

When Brian and Victor start to move from their location, their **sbeo:activityStatus** becomes *sbeo:Evacuating*. Following are the relevant triples to express this information.

**Step 1: Relevant triples in the existing knowledge base.**

:Person2 rdf:type foaf:Person .

:Person7 rdf:type foaf:Person .

:EmergencyEvacuationGroup1 rdf:type sbeo:EmergencyEvacuationGroup .

:Person2 sbeo:hasActivityStatus sbeo:Evacuating .

:Person7 sbeo:hasActivityStatus sbeo:Evacuating .

:EmergencyEvacuationGroup1 sbeo:hasActivityStatus sbeo:Evacuating .

Once they reach at the last space of their provided route (the main exit *sbeo:FE1*, their **sbeo:activityStatus** is updated from *sbeo:Evacuating* to sbeo:*Evacuate*d. This information is given below in the form of triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Person2 rdf:type foaf:Person ;

sbeo:hasActivityStatus sbeo:Evacuating .

:Person7 rdf:type foaf:Person ;

sbeo:hasActivityStatus sbeo:Evacuating .

:EmergencyEvacuationGroup1 rdf:type sbeo:EmergencyEvacuationGroup ;

sbeo:hasActivityStatus sbeo:Evacuating .

**Step 2: Triples those are to be removed from the knowledge base.**

:Person2 sbeo:hasActivityStatus sbeo:Evacuating .

:Person7 sbeo:hasActivityStatus sbeo:Evacuating .

:EmergencyEvacuationGroup1 sbeo:hasActivityStatus sbeo:Evacuating .

**Step 3: Relevant triples those are remained in the knowledge base (only applies to this example)**

:Person2 rdf:type foaf:Person .

:Person7 rdf:type foaf:Person .

:EmergencyEvacuationGroup1 rdf:type sbeo:EmergencyEvacuationGroup .

:Person2 sbeo:hasActivityStatus sbeo:Evacuated .

:Person7 sbeo:hasActivityStatus sbeo:Evacuated .

:EmergencyEvacuationGroup1 sbeo:hasActivityStatus sbeo:Evacuated .

## Autonomous Activity:

### Blind Person:

Being a blind person, Dan is an instance of **sbeo:BlindPerson**. It is expressed in the form of triples which are as following.

:Person4 rdf:type sbeo:BlindPerson ;

foaf:firstName "Dan"^^xsd:string .

Therefore Dan uses a *sbeo:SpecialPurposeDevice1* which is an instance of **sbeo:AlertingDeviceForImpairedPerson** as well as **sbeo:HandheldDevice**. Below are the triples to express this information.

:Person4 sbeo:uses :SpecialPurposeDevice1 .

:SpecialPurposeDevice1 rdf:type sbeo:AlertingDeviceForImpairedPerson ,

sbeo:HandheldDevice .

His **sbeo:meansOfNotification** is a same *sbeo:SpecialPurposeDevice1* which is as follow.

:Person4 sbeo:meansOfNotification :SpecialPurposeDevice1 .

:SPDAudioDescription rdf:type sbeo:AudioDescription .

But his **sbeo:notificationPreference** is **sbeo:AudioDescription** by the fact that he is a **sbeo:BlindPerson** and it is the most appropriate way to inform him. Following is an example of the triples to express this information.

:Person4 sbeo:notificationPreference :SPDAudioDescription .

Furthermore, the route assigned to him must be the *sbeo:SimplestPath* that can be expressed using **sbeo:routePreference** property. It is expressed below in the form of a triple.

:Person6 sbeo:routePreference sbeo:SimpestPath

In this regard, the main **sbeo:BuildingExit** is chosen instead of the **sbeo:EmergencyExit** by the fact that he is **sbeo:familiarWith** the **seas:OfficeBuilding**. Also, **sbeo:EmergencyExit** are **sbeo:connected** to **sbeo:Ramp**, therefore it might not be suitable for him. This piece of information is given below in the form of triples.

:Person6 sbeo:assignedRoute :Route13 .

Once the Dan starts his **sbeo:EmergencyEvacuation** activity, his activity status becomes *sbeo:Evacuating*. It is expressed below in the form of a triple.

:Person4 sbeo:hasActivityStatus sbeo:Evacuating .

The system keeps tracking his **sbeo:hasNavigationalState** that either he is *sbeo:FollowingPath* or *sbeo:DeviatingFromPath.* This is done by matching the momentary location of each person with one of the spaces of provided route. This is expressed in the form of a triple below.

:Person4 sbeo:hasNavigationalState sbeo:FollowingPath .

In case, if a person deviates from one’s path, the number of deviations are calculated using **sbeo:hasXTimeDeviated** property. This is expressed in the form of a triple below.

:Person4 sbeo:hasXTimesDeviated “0”^^xsd:integer .

Various approaches can be adopted if a person deviates from one’s path, such as let one know about one’s deviation and provide that person with the new route which will be calculated from momentary location of a person where one might be.

### Mobility-Impaired Person:

Alice is a mobility-impaired person and uses a motorized wheelchair. Therefore she is an instance of **sbeo:MotorisedWheelchairPerson**. Following are the relevant triples to express this information.

:Person1 rdf:type sbeo:MotorisedWheelchairPerson ;

foaf:firstName "Alice"^^xsd:string .

Consequently, instances of **sbeo:Route**, that consists of **seas:Stairs** will be filtered using **sbeo:excludedFor** property for Alice. Following are the relevant triples to express this information.

**Step 1: Relevant triples in the existing knowledge base.**

:Route8 rdf:type sbeo:Route .

:Route4 rdf:type sbeo:ExitRoute .

:Route8 sbeo:exlcudedFor :Person1 .

:Route4 sbeo:exlcudedFor :Person1 .

As the route preference of Alice is a shortest path, therefore there are two possible paths. One includes an elevator (*:Route14*) while the other has an emergency exit (*:Route15*). Following are the relevant triples to express this information.

:Person1 rdf:type sbeo:MotorisedWheelchairPerson ;

sbeo:routePreference sbeo:ShortestPath .

:Route14 rdf:type sbeo:EmergencyEvacuationRoute ;

sbeo:routeType sbeo:ShortestPath .

:Route15 rdf:type sbeo:EmergencyEvacuationRoute ;

sbeo:routeType sbeo:ShortestPath .

However, *:Route14* is not accessible for the moment by the fact that it is already being used (**sbeo:uses**) by Brian and Victor, and once it will get free, therefore the availability status of elevator becomes ‘Unavailable’. Following are the relevant triples to express this information.

:Person2 sbeo:uses :PV1 .

:Person7 sbeo:uses :PV1 .

:PV1 sbeo:hasAvailabilityStatus sbeo:UnAvailable .

As a result, *:Route15* route is assigned to Alice. This is expressed in the form of triples below.

:Person1 sbeo:assignedRoute :Route14 .

As soon as Alice exits the office space, her activity status becomes evacuated. This is expressed in the form of triples below.

:Person1 rdf:type sbeo:MotorisedWheelchairPerson ;

sbeo:hasActivityStatus sbeo:Evacuated .

### Deaf Person:

Elliot is a deaf person, therefore he becomes an instance of **sbeo:DeafPerson.** This is expressed in the form of triples below.

:Person5 rdf:type sbeo:DeafPerson ;

foaf:firstName "Elliot"^^xsd:string .

On the other hand, he is familiar with the geometry of the building. This is expressed in the form of triples below.

:Person5 sbeo:familiarWith :OfficeBuilding1 .

:OfficeBuilding1 rdf:type seas:OfficeBuilding .

His **sbeo:meansOfNotification** is a mobile phone (*:Phone3*) which is an instance of **sbeo:SmartPhone**. This is expressed in the form of triples below.

:Person5 sbeo:meansOfNotification :Phone3 .

:Phone3 rdf:type sbeo:SmartPhone .

His **sbeo:notificationPreference** is an *:Email* which is an instance of **sbeo:TextualDescription.** This piece of information is given below in the form of triples.

**Step 1: Relevant triples in the existing knowledge base.**

:Email rdf:type sbeo:TextualDescription .

:Person5 sbeo:notificationPreference :Email .

His route preference is a shortest path as well. He is provided with a route of an emergency exit. Following are the triples to express this information.

**Step 1: Relevant triples in the existing knowledge base.**

:Person5 sbeo:routePreference sbeo:ShortestPath .

:Route16 rdf:type sbeo:EmergencyEvacuationRoute ;

sbeo:routeType sbeo:ShortestPath .

:Person5 sbeo:assignedRoute :Route16 .

We have explained a simple scenario for a hazard and route recommendation in a closed multi-storey office building, in which there are various types of persons. The above scenario describes how the Smart Building Evacuation Ontology (SBEO) can be used to perform various functions at the same time. Although it is a basic instantiation of the SBEO, it is easily upgradable according to any specific scenario such as routes in shopping malls to shop concerning the preferences of users or the route to the seating in multi-plex cinemas or stadiums, and so forth.