LAURA architecture situation types

This document presents the complete list of situation types that were used in entitled study **'LAURA architecture: Towards a simpler way of building Situation-Aware and Business-Aware final WSN/IoT Applications'** that is awaiting approval. The proposed situation types are: (i) 'Observed Situation', characterized when a Person indicates his/her interest in monitoring a specific product. (ii) 'Absent Sensor Reading' that describes a situation in which there is a breakdown in communication or data reception for a pre-determined period of time; (iii) 'Exceeding Threshold Situation', when a temperature measurement is beyond the acceptable range for a certain product; (iv) 'Exceeding Safety Distance Situation', when a Person is at a distance from the Container that is far to be considered safe; (v) 'Estimated Time of Arrival (ETA) Greater than Estimated Time-to-Threshold (ETT) Situation', when the estimated time for a Person to reach a Container passes the estimated time that it would take for the Container to reach an unsafe temperature previously defined for that product; (vi) 'All Observers are with ETA Greater than ETT Situation', when all Observers are found in situation (v); (vii) 'Busy Situation', when an Observer indicates that he/she is busy or unavailable to take action in response to situation (vi); (viii) 'Attending Situation', when an Observer indicates that he/she is available to take action in response to the situation (vi). Tables 1 to 7 present the definitions of these types of situations in SCENE.

Table 1
The ***Observing*** situation declaration in SCENE.

Line	Instruction
01	declare **Observing** extends Situation
02	observer: Entity @part
03	observed: Entity @part
04	end
05	rule **Observing**
06	@role(situation)
07	@type(Observing)
08	When
09	obs: RelationalContext(kind="Observation",
10	value.entries["status"] == "ON")
11	RelationalPart(relation == obs, label="observer", observer:
12	entity)
13	RelationalPart(relation == obs, label="observed", observed:
14	entity)
15	then
16	SituationHelper.situationDetected(drools);
17	end

Table 2
The ***AbsentSensorReading*** situation declaration in SCENE.

Line	Instruction
01	declare **AbsentSensorReading** extends **Situation**
02	container: Entity @part
03	end
04	rule **AbsentSensorReading**
05	@role(situation)
06	@type(AbsentSensorReading)
07	when
08	temperature: IntrinsicContext(kind=="temperature")
09	container: Entity(kind=="Container") from
10	temperature.bearer
11	not (
12	ContextValue(context == temperature)
13	over window:time(1m30s)
14	)
15	then
16	SituationHelper.situationDetected(drools);
17	end

Table 3
The ***ExceedingThreshold*** situation declaration in SCENE.

Line	Instruction
01	declare **ExceedingThreshold** extends Situation
02	product: Entity @part
03	end
04	rule **ExceedingThreshold**
05	@role(situation)
06	@type(ExceedingThreshold)
07	when
08	containment: RelationalContext(kind=="Containment")
09	RelationalPart(label=="contained", relation==containment,
10	product: entity)
11	RelationalPart(label=="container", relation==containment,
12	container: entity)
13	temperature: IntrinsicContext(kind=="temperature",
14	bearer == container,
15	value.entries["temperature"] (>=
16	product.attributes["maxThreshold"] || <=
17	product.attributes["minThreshold"]))
18	)
19	then
20	SituationHelper.situationDetected(drools);
21	end

Table 4
The ***ExceedingSafetyDistance*** situation declaration in SCENE.

Line	Instruction
01	declare **ExceedingSafetyDistance** extends Situation
02	observer: Entity @part
03	container Entity @part
04	end
05	rule **ExceedingSafetyDistance**
06	@role(situation)
07	@type(ExceedingSafetyDistance)
08	when
09	obs: RelationalContext(kind="Observation",
10	value.entries["status"] == "ON")
11	RelationalPart(relation == obs, label="observer", observer:
12	entity)
13	RelationalPart(relation == obs, label="observed", product:
14	entity)
15	containment: RelationalContext(kind="Containment",
16	value.entries["status"] == "ON")
17	RelationalPart(relation == containment, label=="contained",
18	entity == product)
19	RelationalPart(relation == containment, label=="container",
20	container: entity)
21	containerLocation: IntrinsicContext(kind=="geolocation",
22	bearer == container)
23	observerLocation: IntrinsicContext(kind=="geolocation", bearer
24	== observer)
25	eval(distance (containerLocation, observerLocation) > 50km)
26	then
27	SituationHelper.situationDetected(drools);
28	end

Table 5
The ***ObserverETAGreaterThanETT*** situation declaration in SCENE.

Line	Instruction
01	declare **ObserverETAGreaterThanETT** extends Situation
02	observer: Entity @part
03	container: Entity @part
04	product: Entity @part
05	end
06	rule **ObserverETAGreaterThanETT**
07	@role(situation)
08	@type(**ObserverETAGreaterThanETT**)
09	when
10	observation: RelationalContext(kind=="Observation",
11	value.entries["status"] != "BUSY")
12	RelationalPart(relation == observation, label=="observer",
13	observer: entity)
14	RelationalPart(relation == observation, label=="observed",
15	product: entity)
16	containment: RelationalContext(kind="Containment",
17	value.entries["status"] == "ON")
18	RelationalPart(relation == containment, label=="contained",
19	entity == product)
20	RelationalPart(relation == containment, label=="container",
21	container: entity)
22	ett: IntrinsicContext(kind=="EstimatedTimeToThreshold",
23	bearer == product)
24	eta: RelationalContext(kind=="EstimatedTimeOfArrival",
25	value.entries["eta"] > ett.value.entries["ett"])
26	RelationalPart(relation == eta, label=="person", entity ==
27	observer)
28	RelationalPart(relation == eta, label=="container", entity ==
29	container)
30	then
31	SituationHelper.situationDetected(drools);
32	end

Table 6
The ***AllObserversETAGreaterThanETT*** situation declaration in SCENE.

Line	Instruction
01	declare **AllObserversETAGreaterThanETT** extends Situation
02	product: Entity @part
03	observers: List @snapshot
04	end
05	rule **AllObserversETAGreaterThanETT**
06	@role(situation)
07	@type(**AllObserversETAGreaterThanETT**)
08	when
09	Observing(person: observer, product: observed)
10	forall (
11	Observing(person == observer,
12	product == observed,
13	active)
14	ObserverETAGreaterThanETT(this.person == person,
15	active)
16	)
17	observers: ArrayList() from collect(Observing(
18	product == observed,
19	active))
20	then
21	SituationHelper.situationDetected(drools);
22	end

Table 7
The ***Busy*** situation declaration in SCENE.

Line	Instruction
01	declare **Busy** extends Situation
02	person: Entity @part
03	product: Entity @part
04	end
05	rule **Busy**
06	@role(situation)
07	@type(Busy)
08	When
09	ObserverETAGreaterThanETT(person: observer, product:
10	product)
11	observation: RelationalContext(kind=="Observation",
12	value.entries["status"] == "BUSY")
13	RelationalPart(relation == observation, label=="observer",
14	observer == entity)
15	RelationalPart(relation == observation, label=="observed",
16	product == entity)
17	then
18	SituationHelper.situationDetected(drools);
19	end

Table 8
The **Attending** situation declaration in SCENE.

Line	Instruction
01	declare **Attending** extends Situation
02	person: Entity @part
03	product: Entity @part
04	end
05	rule **Attending**
06	@role(situation)
07	@type(**Attending**)
08	when
09	ObserverETAGreaterThanETT(person: observer, product:
10	product)
11	observation: RelationalContext(kind=="Observation",
12	value.entries["status"] == "ATTENDING")
13	RelationalPart(relation == observation, label=="observer",
14	person == entity)
15	RelationalPart(relation == observation, label=="observed",
16	product == entity)
17	then
18	SituationHelper.situationDetected(drools);
19	end