Semantic Reasoning for Intelligent Emergency Response Applications

Anna Hristoskova, Femke Ongenae and Filip De Turck
Department of Information Technology
Internet Based Communication Networks and Services research group
Ghent University - iMinds
Gaston Crommenlaan 8, bus 201, 9050 Ghent, Belgium
Email: Anna.Hristoskova@intec.ugent.be

***Abstract*—Emergency response applications require the processing of large amounts of data, generated by a diverse set of sensors and devices, in order to provide for an accurate and concise view of the situation at hand. The adoption of semantic technologies allows for the definition of a formal domain model and intelligent data processing and reasoning on this model based on generated device and sensor measurements.**

This paper presents a novel approach to emergency response applications, such as fire fighting, integrating a formal semantic domain model into an event-based decision support system, which supports reasoning on this model. The developed model consists of several generic ontologies describing concepts and properties which can be applied to diverse context-aware applications. These are extended with emergency response specific ontologies. Additionally, inference on the model performed by a reasoning engine is dynamically synchronized with the rest of the architectural components. This allows to automatically trigger events based on predefined conditions. The proposed ontology and developed reasoning methodology is validated on two scenarios, i.e. (i) the construction of an emergency response incident and corresponding scenario and (ii) monitoring of the state of a fire fighter during an emergency response.

I. Introduction

Emergency responders, such as fire fighters, regularly face large amounts of data generated by a diverse set of sensors and devices. These need to be processed in a timely manner in order to form astute decisions during a disaster. An emerging trend [1] in such settings is the development of context-aware decision support systems able to provide an accurate and concise view of the situation at hand. Relevant information, captured from various devices and sensors, should be pushed pro-actively and presented in a context-aware way [2] in order to support the situational awareness of the actors involved.

Current emergency response research focuses on decision support systems [3], [4] and crisis simulation environments [5]. However, the underlying databases provide limited information processing and reasoning. In our approach, semantic technologies are adopted, enabling the formal definition of the domain concepts and their properties [6] in an emergency response ontology. This supports intelligent reasoning on the available data inferring valuable insights on the current context.

The reasoning framework proposed in this paper seamlessly integrates a domain-specific semantic model into a decision support system for emergency response by the fire department in the context of the ASTUTE[1] project. The novelty of the described approach is twofold. First, the developed semantic model is defined by means of several generic ontologies which can be used to describe diverse context-aware application domains. These are extended with emergency response specific ontologies. Second, the overall architecture consists of the seamless combination of a semantic reasoner and an event-based decision support system. Incoming real-time data from device and sensor measurements during an emergency is updated into the semantic domain model. The reasoner automatically derives new knowledge from this formal emergency response definition. Based on the inferred context, the event-based decision support system triggers alarms which are forwarded to the right fire fighting units.

II. Related Work

Ongoing development of emergency response applications targets issues such as distributed communication between mobile devices, simulation environments for training purposes, decision support systems processing events from numerous devices, sensors, social media, and formal domain modelling.

Raven [7] supports the use of smart phones for collaborative disaster data collection and sharing through an exchange of the database and corresponding schema. Its current interface tracks lost and found people based on database records. Similarly, SocEDA [8], [9] enables the exchange of social network data between heterogeneous services through the distributed interaction of several event processing engines. An emergency response scenario during a nuclear disaster simulates virtual events from the involved partners.

The suite of adaptive search methods applied by REScUE [3] constructs a near-optimal plan consisting of an associated response team, including equipment and vehicles, using information on the location and capabilities of the available resources. In addition, emergency response agents at the scene of the event report to the decision support system information on casualties and urgent tasks. REScUE's agent-based simulation environment, STORMI, evaluates the response of the emergency services to hypothetical major incidents. Simulation technology for crisis management and training is also

[1]ASTUTE is a large EU project (www.astute-project.eu) which focuses on the development of pro-active decision support for data intensive environments. The approach is being verified in several different industrial demonstrators e.g. avionics, emergency management and others.

supported by the INDIGO FP7 EU project [5] which uses a whiteboard to share information between the crisis centre and the mobile devices in the field.

Incidone [4] provides incident info, loose dynamic action lists, suggestions based on data, and composition of tasks to coast guard watch officers in a centralized command centre. All knowledge of the area is collected from contacts outside or automated systems like the Automatic Identification System. Additionally, plans in the form of hierarchical to-do lists are used to coordinate the actions taken to resolve incidents.

WeKnowIT [10], [11] extracts information on emergency response scenarios (e.g. flooding, fire) from user-generated sources such as emergency response workers at the scene or the general public observing. This information is geo-located, either from image metadata or through a textual or visual analysis, and displayed on a map. An emergency alert service [12] informs social contacts and public authorities about the emergency situation. Events and user interactions are represented by means of the WeKnowIt core ontology, CURIO [2], which defines resources for user generated content.

The FP7 EU project PRONTO [13], [14] focuses on event recognition for intelligent resource management. Its semantic data store has an ontology of events defining the status of the system. These events are gathered from various kinds of sources, such as hardware devices (e.g. GPS), user interactions with the system, audio and video data streams. After aggregation, relevant information is filtered out for the users.

The ontology described in [15] presents a model for the organization of dynamic data for emergency response developed within the RGI-239 project 'Geographical Data Infrastructure for Disaster Management' (GDI4DM). It applies ontologies to resolve the semantic interoperability inherent in emergency management. The model is derived from the organization of emergency response in the Netherlands investigating the information flow from processes performed by first responders such as fire brigade, paramedics, police and municipality. It captures the type of disaster, the involvement of response sectors including their locations, and the consequences for people, animals and infrastructure. The main objective is the extraction and processing of the information from spatial data sets and its distribution to the different response units.

The main contribution of the described approach in this paper is the enrichment of an event-based decision support system with semantic reasoning on a formal domain model. The model consists of a combination of several generic ontologies used to describe diverse context-aware application domains extended with emergency response specific features. During an emergency the model is updated with incoming real-time data from devices and sensors. A reasoner automatically infers new knowledge from the updated emergency response context based on which the event-based system triggers events and alarms forwarded to the right emergency response units.

III. Integrated Reasoning Framework

This paper proposes a layered approach combining the formal definition of a domain model using an ontology, semantic reasoning on this model and the triggering of events based on new, enriched information inferred from the ontology. The advantage is the distinction between the reasoning on the model from the application-specific actions.

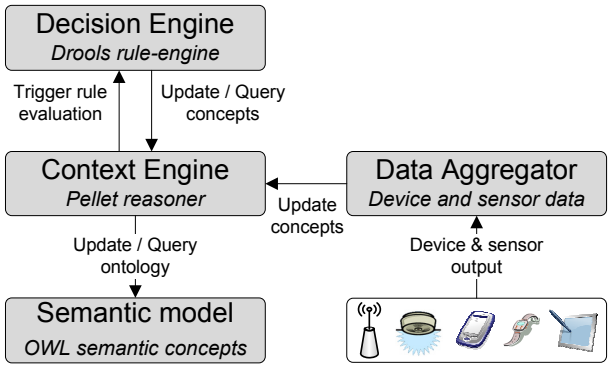

Fig. 1. Reasoning framework layers: illustration of the importance of the semantic model within the application.

Figure 1 presents an overview of the architectural layers of the ASTUTE framework. The first layer is the *Semantic model* which contains the Emergency Response Ontology described in Section IV. The layer above, the *Context Engine*, encapsulates the translation of the semantic concepts into Java Beans and uses Pellet [16] to reason on the model. These objects are queried by the *Decision Engine* which utilizes the Drools rule engine [17] in order to trigger events and alarms intended for the emergency responders. The Java Bean translation, in Section V, enables the transparent use of an actual semantic model by Drools resulting in the triggering of rules on the updated objects in a timely manner. Finally, the *Data Aggregator* is responsible for capturing data from devices and sensors and formatting it as defined by the *Semantic model* using the encapsulation of the *Context Engine*.

The main purpose of the *Context Engine* is the semantic reasoning on the domain model, which generates knowledge flow into the system through inference from new data. The *Decision Engine* captures application knowledge in the form of rules in order to determine which information needs to be sent to whom at what moment. It relies on the *Context Engine* for delivering the interpreted raw context data.

IV. Emergency Response Ontology

In order to develop intelligent emergency response applications which are able to interpret the meaning and adequately filter the relevant information out of the huge amount of heterogeneous data provided about an emergency situation, an Emergency Response Ontology was developed.

A. Ontology Development Methodology

The Emergency Response Ontology was designed by ontology engineers in close collaboration with project stakeholders such as industry professionals who have a long track record of developing ICT solutions for emergency management. These companies have a broad knowledge of the data being exchanged during an emergency situation and have a direct link to the domain experts working in the field, e.g., fire fighters. This close collaboration ensures that the information in the ontology accurately and completely reflects the daily

[2]http://socsem.open.ac.uk/ontologies/curio/

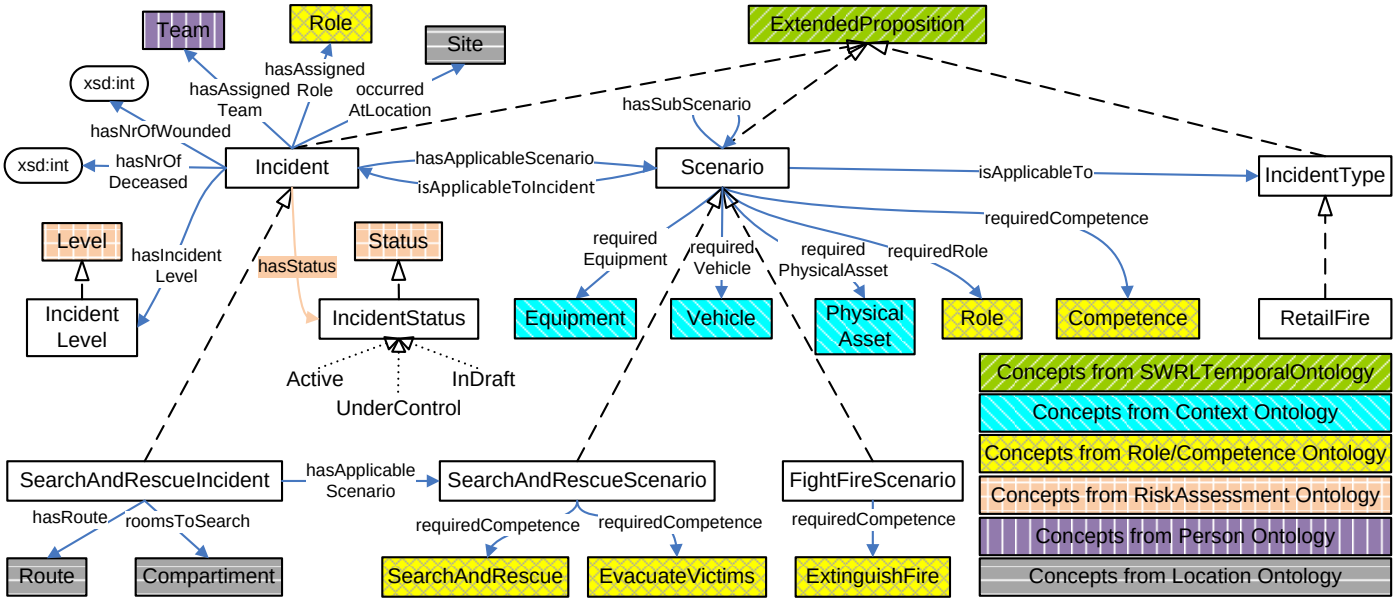

Fig. 2. Graph-based visualization of the *Low-Level Emergency Plan Ontology*.

work practices of the domain experts who will be using the applications built based on this semantic model. Several methods were employed to involve these partners in each step of the ontology life cycle without having to construct the ontology themselves. The five widely accepted stages for building an ontology [18] are *Specification*, *Conceptualization*, *Formalization*, *Implementation* and *Maintenance*.

The aim of the *Specification* stage is to define the scope of the ontology. In order to achieve this various scenarios were developed detailing the use of the emergency response system. These help develop a detailed and shared understanding of the context and activities of the future system users.

In the second *Conceptualization* stage, a conceptual model of the ontology is constructed. This consisted of the extraction of the different concepts and properties of the emergency response domain from the scenarios. Large amount of the modelled knowledge is also applied within other context-aware application domains, such as capturing data on devices, sensors, people, companies, locations. Therefore, this knowledge was modelled in separate ontologies, called *High-Level* or generic ontologies, such that they can easily be re-used for other applications. *Low-Level* ontologies extend these *High-Level* ontologies with concepts and properties which are specific for the emergency response domain. An investigation of existing ontologies was also performed to evaluate if these could be re-used and how they should be integrated. In order to iteratively discuss and fine-tune the ontology under development with the different stakeholders, graph-based visualizations were constructed. An example of such a graph is shown in Figure 2. The squares represent concepts, the blue arrows depict the relations between them, the dashed arrows represent subclass relationships and the dotted arrows indicate instances of a certain concept.

The *Formalization* stage translates the conceptual model into a formal model by adding axioms and rules that restrict its possible interpretations, e.g., defining which competences a certain role has or are needed to perform a task, which sensor observations are valid. The properties of the relations, domain and range, were discussed with the stakeholders by using the graph-based visualizations. However, in order to define more complex definitions, an Excel-sheet was used, which allows stakeholders to informally express and discuss the restrictions which are needed, by defining domain conditions, arguments, effects and relations to other restrictions in a natural language. This frees the stakeholders from learning the required formal language constructs. The natural language descriptions are then translated to formal axioms and SWRL rules [19] by the ontology engineers in the *Implementation* stage.

During the *Implementation* stage, the conceptual graphs and rules are translated into an ontology. The Protégé editor [20] was used to develop the ontology in the Ontology Web Language (OWL) [21] and express the rules on the concept properties in the SWRL rule language. The ontology consistency and classification was checked by the Pellet Reasoner [16]. The resulting ontology is discussed in the following section.

The *Maintenance* stage consists of the continuous evaluation, update and correction of the ontology. This work is ongoing within the ASTUTE project. The integrated reasoning framework is currently under development. Its aim is to demonstrate the development of intelligent emergency response applications supporting fire fighters during an emergency situation on top of this ontology. As a proof of concept an application is being developed which supports fire fighters in a search and rescue situation, allowing them to rapidly find and rescue victims while monitoring their own safety and health. It also allows the commander to track the progress and condition of his team. The development of the application allows gaining significant insights into the completeness and usability of the ontology. Moreover, stakeholders can experiment, criticize, discuss and evaluate the application which will give input on the correctness and applicability of the ontology.

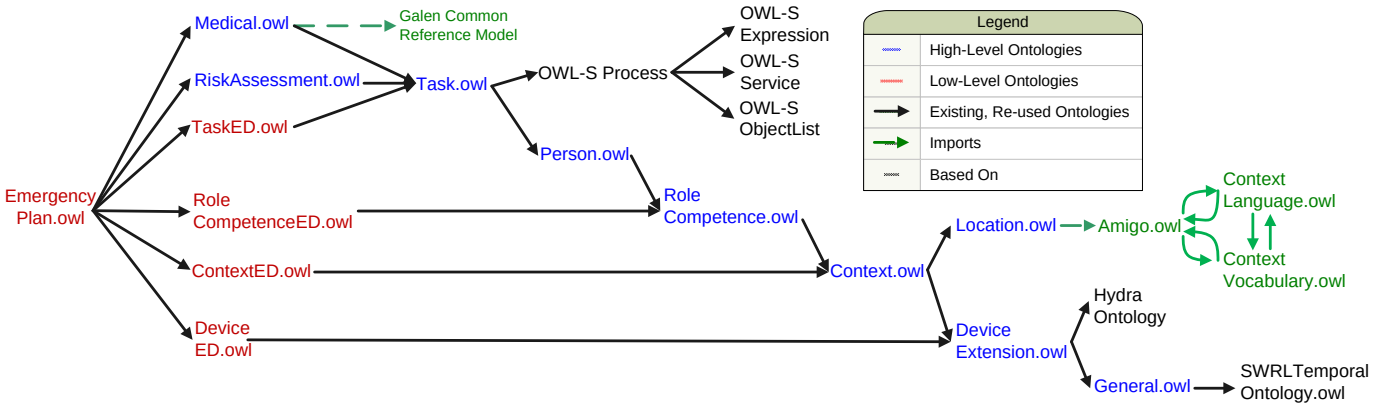

Fig. 3. Import schema of the *Emergency Response Ontology*.

B. The Resulting High-Level and Low-Level Emergency Response Ontologies

Within ASTUTE 8 *High-Level* ontologies and 6 *Low-Level* ontologies were developed. Figure 3 visualizes how they are related to each other and which existing ontologies they are based on or import.

General High-Level Ontology: Classes, properties and axioms that are generic across all knowledge domains are described by this upper ontology. An upper ontology, also known as a top-level ontology or foundation ontology, is an ontology which describes very general concepts that are the same across all knowledge domains [22]. Common examples are the *SWRLTemporalOntology* [23], WordNet [24], Cyc[3] and the Suggested Upper Merged Ontology (SUMO) [25]. Upper ontologies have however received a lot of critique. It is difficult to capture the concepts in such a way that they can easily be re-used across different domains without modification. Specific domains often require small alterations to the concepts defined in the upper ontologies. Sometimes the concepts are too generic to be usefully re-used. A lot of the concepts defined in the upper ontologies clash with concepts defined in the ontologies we already wanted to import. However, it is important to model time in the ontology, e.g., tasks that need to be performed before a certain point in time, for logging purposes or work processes that are adjusted according to the time of day. Consequently, The *General High-Level Ontology*, imports the *SWRLTemporalOntology* ontology and extends it with additional concepts modelling the current temporal context, e.g., season, shift or holidays.

Device Extension High-Level Ontology: This ontology describes concepts and properties related to devices, such as the device status and events from devices defining values with their respective units. There exist a whole plethora of ontologies to describe devices, sensors and sensor networks. A comprehensive list can be found in Section 4 of the W3C Semantic Sensor Network XG Final Report [26] of the W3C Semantic Sensor Network Incubator group. For this research, the Hydra ontology [27] was imported to define the actual devices. The *Device Extension High-Level Ontology* allows interpreting and attaching meaning to the data values monitored by the sensors or devices using an observation pattern. A data value monitored by a system is modelled as an `Observation`. Rules added to the ontology allow detecting specific phenomena in these observations, e.g., oxygen level is zero. These phenomena are modeled as `Symptom` concepts. Classification axioms allow reclassifying these symptoms as `Faults`, e.g., oxygen in the room is too low. Each fault can then again be reclassified as having a `Solution`.

Location High-Level Ontology: A formal description of locations including type of buildings, rooms, floors, passages, GPS coordinates and reference systems is provided by this ontology. It is based on the Amigo [28] ontology.

Context High-Level Ontology: Herein additional context information not captured by the *Device Extension* and *Location High-Level Ontologies* is described, e.g., physical assets such as vehicles, furniture and equipment. Moreover, this ontology also defines relations, restrictions and properties that relate the concepts from both ontologies, e.g., associating a location with its temperature or oxygen level based on the measurements of sensors in the neighbourhood of this location.

Person High-Level Ontology: It defines the profile information of people such as their contact information and the organizations they work for. A `Profile` consists of a `Basic` and a `Risk Profile`. The `Basic` profile models the important administrative (birth date, sex), psychological (easily stressed) and sociological (nationality and language) information. This information is inputted into the system or extracted from documents such as the personnel file. The `Risk` profile is defined by classification axioms and rules. This allows a reasoner to automatically obtain the `Risk` profile of a person by reasoning on the information in the `Basic` profile. The *Person High-Level Ontology* also defines relations which allow associating a person with concepts from the previously discussed ontologies, e.g., to indicate the location of the person or which items and devices he/she owns. Finally, a person can also have a status, e.g., free, busy, missing or stressed. Axioms are defined automatically extracting this status based on the rest of the information in the ontology.

Role & Competence High-Level Ontology: In order to provide support to the people with the tasks they need to perform during an emergency situation, it is important to define their competences and roles. A lot of research has been done on how roles should be represented in knowledge representation

[3]http://www.cyc.com/

models [29], [30]. In ontology engineering a distinction is often made between `Entities`, i.e., things that are, from `Events`, i.e., things that happen. It is not obvious how this division admits `Roles`, i.e., things that are, but only in the context of things that happen. Research has produced three basic approaches for representing roles:

- A role can be represented as a label assigned to a participant in an event [31]. This approach is simple, but it fails to make the distinction between roles and entities and makes it difficult to add characteristics to roles.
- Roles are differentiated from entities, but both concepts are then combined into one single hierarchy [32], [33]. This combination can be done in two ways. The roles can be subtypes of entities which is problematic when entities of different types can play the same role. The roles can also be super-types of entities which also leads to modelling problems related to the dynamicity of roles.
- A role is represented as an "adjunct instance" of an entity [29]. This is a distinct instance of a role class that is coupled with the instance of an entity. The role instance does not exist independent of that entity.

The latter approach was adapted in this project. In the *Role & Competence High-Level Ontology* roles are types independent of entities. They thus have two separate hierarchies. An instance of a role is played by an instance of an entity. Thus, every instance of a role exists along with an instance of an entity. Our representation extends further on this approach by also including competences. Each role is defined by its competences through classification axioms. This allows writing algorithms that find the most appropriate staff members to fulfil a task based on the required competences. Roles and competences can also have characteristics (properties) in our representation, e.g., the department a person works on when he/she has a certain role. Each person is associated with competences and roles through five relationships:

- `hasFunction`: primary role of this person, i.e., the role for which this person was primarily hired.
- `hasRole`: models all the roles this person can have, e.g., the fire truck driver who is also a fire fighter.
- `hasCurrentRole`: role the person is currently fulfilling within the emergency response setting. If this relation is not instantiated, it is assumed that the current role of the person is his or her function.
- `hasDiplomaCompetence`: extra competences this person has acquired by following courses.
- `hasExperienceCompetence`: extra competences this person has acquired through experience.

Task High-Level Ontology: The emergency response ontology requires the modelling of process workflows executed during emergency situation such as fire fighting. OWL-S [4] is an ontology for describing Semantic Web Services. Its process model allows describing process workflows, how processes can be mapped on each other based on their inputs and outputs, which conditions need to be fulfilled to execute the process and which effects the execution of the process has on the environment and the context. This way workflows can be constructed that start from particular input and context and reach a specified effect and result by combining various tasks. The *Task High-Level Ontology* extends this OWL-S Process ontology by introducing the `Task` concept, which is a subclass of `Process`, divided into `Planned` and `Unplanned` tasks. Each task has also an associated `Status`, e.g., `Assigned`, `Finished` or `Active`, `Priority`, `Location` at which it is preferably executed and `Competences` needed to execute it. This `Task` concept is used to model the various emergency response tasks along with the required competences. Consider, for example, assigning the task of searching a victim in a fire. Searching a victim is modelled as an `Unplanned Task`. For each type of a task it can then be specified which competences or roles are needed to handle it, e.g., search and rescue operations can only be performed by fire fighters. For each type of task, its preconditions (a person is missing), its input (the location where the victim was last seen), its output (the location of the missing person), and its effects (the commander is alerted that the victim was found and notified of his/her location), are modelled by using concepts from the OWL-S Process ontology.

Medical High-Level Ontology: For the emergency response domain a formal definition of medical knowledge is required which allows monitoring the medical condition, e.g., heart rate or body temperature, of the various rescue workers. A well-known eHealth ontology is the *Galen Common Reference Model*[5] which is developed as a clinical terminology. Along with modelling clinical categories it contains sufficient information on those categories allowing for their automatic classification. The *Galen Common Reference Model* especially avoids adding too many axioms to the ontology that constrain the possible interpretations of a concept, unless there is an agreement about the constraint, e.g., an ulcer located in the stomach is a stomach ulcer. As importing the whole ontology would significantly slow down the reasoning, it was decided to only import the concepts of the *Galen Common Reference Model* which are needed in this project. These concepts are preceded by the *galen* namespace prefix. This way, the concepts and the ontology can be easily mapped back to the original *Galen Common Reference Model*. The *Medical High-Level Ontology* adds axioms and constraints to this imported terminology that express relations between the medical knowledge and concepts in the other ontologies. For example, it defines the `hasMedicalParameter` property which associates each person with his/her medical parameters.

Device, Context, Role & Competence and Task Emergency Demonstrator (ED) Low-Level Ontologies: These ontologies extend the respective *High-Level* ontologies with concepts, restrictions and relations specific for the emergency response domain. For instance, these ontologies model specific equipment (e.g. fire extinguishers), vehicles (e.g. fire trucks), roles and their mapping on competences (e.g. fire fighter and its competences such as extinguishing fire and performing search and rescue missions), devices (e.g. fire alarm control panels) and tasks (e.g. extinguishing fires).

[4] http://www.w3.org/Submission/OWL-S/

[5] http://www.opengalen.org/index.html

Risk Assessment Low-Level Ontology: This ontology models the various risks associated with particular locations or physical assets, i.e., equipment and vehicles. For instance, it is possible to model the presence of explosives within a particular space. Each risk is also associated with the protective measures which should be employed to mediate or diminish the risk. For example, in case there are explosives in a certain space, the fire wall should be protected and a safety distance should be maintained. Finally, each risk is also associated with a level indicating how severe the risk is within the current context.

Emergency Plan Low-Level Ontology: The most vital ontology is the *Emergency Plan Low-Level Ontology*, visualized in Figure 2, which combines the concepts from the rest of the ontologies in defining the specific emergency incident and corresponding scenario. Therein, depending on the scenario, e.g., search and rescue, the required team having specific roles and competences is assigned.

V. Automatic Java Beans Generation From Semantic Concepts

The adoption of a semantic domain model usually requires the configuration of a reasoner such as Pellet and the manual encapsulation of the necessary concepts into Java classes in order to update and query their properties in a more general domain independent way. This manual work may be feasible for limited models but for larger use cases such as an emergency dispatching scenario the amount of manual work becomes difficult to maintain. Apart from the necessary testing, one needs to manually update the Java classes each time the ontology changes in order to keep the system synchronized with the domain model.

We automated this process through the creation of a Code Generator [34] that automatically generates Java classes from semantic concepts translating their properties into class methods. For example, the following methods are automatically generated for the property `hasCommander(FireFighter, Commander)` which specifies that a `FireFighter` has a `Commander`:

```
public interface FireFighter extends Role {
    /*
     * Property http://localhost/ASTUTE.owl#hasCommander
     */

    // Gets all values for the hasCommander property.
    Collection<? extends Commander> getAllCommander();

    // Gets the value for the hasCommander property.
    Commander getCommander();

    // Checks if it has a hasCommander property value.
    boolean hasCommander();

    // Adds a hasCommander property value.
    void addCommander(Commander newCommander);

    // Removes a hasCommander property value.
    void removeCommander(Commander oldCommander);

    // Sets a hasCommander property value.
    void setCommander(Commander newCommander);
    ...
}
```

The result is the automatic translation of the semantic domain concepts into Java Beans that are used by Drools just like regular Beans. In order for Drools rules to retrieve the specific commander name value, one should query its data property value `hasName` and thus use: `FireFighter(commander.getName())`. On the other hand, as data properties specify value type definitions, it is possible to use them in the following way: `Commander(name)`.

Consequently these objects are consumed by the rest of the application just like normal objects with the exception that the underlying model is updated together with the object updates. Additionally, inference on the model performed by the semantic reasoner in the **Context Engine** is automatically synchronized with the **Decision Engine** resulting in the triggering of events by Drools based on scenario specific conditions.

VI. Illustrative Example to an Emergency Response Application

In order to validate the proposed ontology and developed reasoning methodology this section describes two cases, i.e. (i) the construction of an emergency response incident and corresponding scenario and (ii) the monitoring of the state of a fire fighter during an emergency response.

A. Initialization of an Emergency Incident

As mentioned in Section IV-B, Figure 2 presents the *Emergency Plan Low-Level Ontology*, which specifies an emergency incident and corresponding scenario through a fusion of the rest of the ontologies. The particular scenario description, e.g., search and rescue, enables the selection of the required team having specific roles and competences. Following is a description of how this assignment is accomplished.

As visualized by Figure 2 one is able to specify an emergency `Incident` at a `Location` and assign a `Team` and `Roles` to it. Additionally, it is possible to link to it a `Scenario` describing the required assets such as `Equipment`, `Vehicles`, `Devices`, `Roles` and `Competences` to accomplish tasks. Based on this general `Incident` definition one can supply the assigned `Team` with the required assets.

However if the incident at hand is more specific such as a search and rescue, it should be possible to link a predefined search and rescue scenario to it which poses particular requirements. For example, the following ontology definition specifies a restriction on a `SearchAndRescueIncident` bounding it to a `SearchAndRescueScenario`:

```
class SearchAndRescueIncident
        Superclasses: Incident
                      hasApplicableScenario
                        some SearchAndRescueScenario
```

This `SearchAndRescueScenario` is defined as requiring specific competences such as `EvacuateVictims` and `SearchAndRescue` as follows:

```
class SearchAndRescueScenario
        Superclasses: Scenario
                      requiredCompetence some
                        (EvacuateVictims and SearchAndRescue)
```

Based on this specification one can select a qualified team with roles supporting these competences. Figure 4

presents a partial description of the *Role & Competence Low-Level Ontology* where fire fighter `Roles` are linked to `Competences`. Using this information a matching `Role` for the `EvacuateVictims` and `SearchAndRescue` `Competences` is a `ReconFireFighter`.

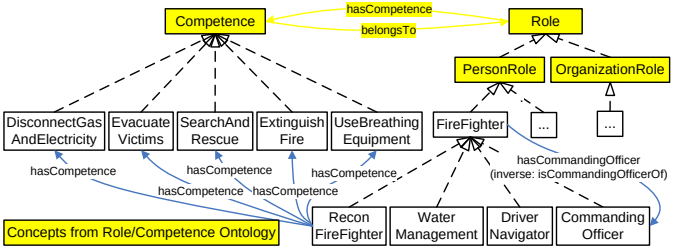

Fig. 4. Competences assigned to specific team roles.

This is accomplished through the definition of the following SWRL rule stating that if the `Scenario` requires these `Competences`, then the `Incident` should dispose of the specific `Roles` corresponding with such `Competences`:

```
IF
        Incident hasApplicableScenario Scenario AND
        Scenario requiredCompetence Competence AND
        Role hasCompetence Competence
THEN
        Incident hasAssignedRole Role
```

A similar role assignment is accomplished through an alternative rule where a specific `Incident` has an assigned `Team` whose participants have specific Roles:

```
IF
        Incident hasAssignedTeam Team AND
        Person hasCurrentRole Role AND
        Team hasTeamParticipant Person
THEN
        Incident hasAssignedRole Role
```

The information on the emergency incident and scenario provides a description of the emergency situation, specifies the available resources and supports tracking of the progress of the emergency scenario. The description is further used to assign `Tasks` requiring specific `Competences` to people disposing of these `Competences`.

B. Tracking of a Firefighter's State

In order to validate the proposed approach, a second scenario is designed that captures the state of a person, namely a fire fighter under stress walking through a building on fire. It presents the description of a fire fighter who can be a person having several properties such as a location, activities, e.g., extinguishing fire, and medical measurements, e.g., temperature, heart rate and oxygen level. Due to the specification of several types of context, such as physical, task and medical, one can define the criticality of a task or the level of a medical measurement. For example, the activity `ExtinguishingFire` that is performed by fire fighters is defined as a `Task` with `High Criticality` as specified by the necessary condition below:

```
class ExtinguishingFire
        Superclasses: Task
                      hasCriticality value High
```

Using these definitions one can define SWRL rules inferring the thresholds for high temperature or heart rate. The following rule defines that a `BodyTemperature` above 38 degrees is a `HighBodyTemperature`:

```
IF
        BodyTemperature greaterThan 38
THEN
        BodyTemperature is a HighBodyTemperature
```

We could also specify personalized thresholds per person where one can define a maximum heart rate specific for each person. The following rule defines a person `HeartRate` higher that his personal defined maximum as a `HighHeartRate`:

```
IF
        Person hasMedicalParameter HeartRate_1 AND
        Person hasMaxHeartRate HeartRate_2 AND
        HeartRate_1 greaterThan (HeartRate_2 multiply 0.8)
THEN
        HeartRate_1 is a HighHeartRate
```

The results from inferring these rules are combined into additional rules. For example, we can define the following rule that expresses that a person who performs a highly `Critical Task` while having `HighHeartRate` and `HighBodyTemperature` and is located in a `Room` with `HighRoomTemperature` is `Stressed`:

```
IF
        Activity hasCriticality High AND
        Person hasActivity Activity AND
        Person hasBodyTemperature BodyTemperature AND
        Person hasHeartRate HeartRate AND
        HeartRate is a HighHeartRate AND
        BodyTemperature is a HighBodyTemperature AND
        Person hasLocation Room AND
        Room hasRoomTemperature RoomTemperature AND
        RoomTemperature is a HighRoomTemperature
THEN
        Person isStressed true
```

These rules enable the tracking of a fire fighter's state during the fire fighting scenario. The environmental sensors sending various measurements, such as temperature, location and heart rate, register these values via the **Data Aggregator** in the **Context Engine**. With each new update, the **Context Engine** fires the rules in the **Decision Engine**. If the following Drools rule is defined that alerts the commander that the fire fighter is stressed, it will be triggered evaluating the condition of the fire fighter.

```
rule "Firefighter is stressed"
no-loop
  when
    $p : Person(isStressed==true),
    $f : Person.getCurrentRole()
  then
    $hmi.sendMessage($f.getCommander().getName(),
    "Firefighter " + $f.getName() + " is stressed", "ALERT");
end
```

The moment the **Context Engine** receives a new update inferring that the fire fighter is actually stressed because of

for instance an elevated heart rate, the **Decision Engine** fires this rule and the commander is alerted of the state of his team member.

VII. Conclusion

This paper presents a novel approach integrating a semantic reasoning engine into an event-based system supporting the use of a formal domain definition. An Emergency Response ontology is divided into several application-independent ontologies, which are extended with emergency response specific ontologies. Additionally, inference on the model performed by the semantic reasoner is automatically synchronized with the rest of the architectural components resulting in the triggering of events based on scenario specific conditions.

Acknowledgment

A. Hristoskova would like to thank the Special Research Fund of Ghent University (BOF) for financial support through her PhD grant. F. Ongenae would like to thank the Flemish Government Agency for Innovation by Science and Technology (IWT) for financial support through her Ph.D. grant. This work is funded by the project ASTUTE from the ARTEMIS Joint Undertaking, Grant agreement no.: 269334.

References

[1] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *6th International Conference on Ubiquitous Computing, Workshop on Advanced Context Modelling, Reasoning and Management*, 2004, pp. 31–41.

[2] E. Tsiporkova, T. Tourwé, N. González-Deleito, and A. Hristoskova, "Ontology-driven multimodal interface design for an emergency response application," in *9th International Conference on Information Systems for Crisis Response and Management*, 2012.

[3] G. Coates, G. Hawe, D. Wilson, and R. Crouch, "Adaptive co-ordinated emergency response to rapidly evolving large-scale unprecedented events (rescue)," in *8th International Conference on Information Systems for Crisis Response and Management*, 2011.

[4] B. Lijnse, J. M. Jansen, and R. Plasmeijer, "Incidone: A task-oriented incident coordination tool," in *9th International Conference on Information Systems for Crisis Response and Management*, 2012.

[5] *Indigo, Crisis Management Solutions*, http://indigo.diginext.fr/, 2012.

[6] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[7] N. Palmer, R. Kemp, T. Kielmann, and H. Bal, "Raven: Using smartphones for collaborative disaster data collection," in *9th International Conference on Information Systems for Crisis Response and Management*, 2012.

[8] *SocEDA, SOCial Event Driven Architecture*, http://research.petalslink.org/display/soceda/SocEDA+Overview/, 2012.

[9] F. Paraiso, G. Hermosillo, R. Rouvoy, P. Merle, L. Seinturier *et al.*, "A middleware platform to federate complex event processing," in *16th IEEE International EDOC Conference*, 2012.

[10] *WeKnowIt*, http://www.weknowit.eu/, 2012.

[11] S. Diplaris, A. Sonnenbichler, T. Kaczanowski, P. Mylonas, A. Scherp, M. Janik, S. Papadopoulos, M. Ovelgoenne, and Y. Kompatsiaris, "Emerging, collective intelligence for personal, organisational and social use," *Next Generation Data Technologies for Collective Computational Intelligence*, pp. 527–573, 2011.

[12] M. Ovelgonne, A. Sonnenbichler, and A. Geyer-Schulz, "Social emergency alert service-a location-based privacy-aware personal safety service," in *2010 Fourth International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST)*. IEEE, 2010, pp. 84–89.

[13] *Pronto, Event Recognition for Intelligent Resource Management*, http://www.ict-pronto.org/, 2012.

[14] M. Moi and R. Marterer, "An architecture for distributed, event-driven systems to collect and analyze data in emergency operations and training exercises," in *8th International Conference on Information Systems for Crisis Response and Management*, 2012.

[15] Z. Fan and S. Zlatanova, "Exploring ontologies for semantic interoperability of data in emergency response," *Applied Geomatics*, vol. 3, no. 2, pp. 109–122, 2011.

[16] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[17] M. Bali, *Drools JBoss Rules 5.0 Developer's Guide*. Packt Publishing, Limited, 2009.

[18] H. S. Pinto and J. P. Martins, "Ontologies: how can they be built?" *Knowledge and Information Systems*, vol. 6, no. 4, pp. 441–464, 2004.

[19] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, http://www.w3.org/Submission/SWRL/, 2004.

[20] H. Knublauch, R. Fergerson, N. Noy, and M. Musen, "The protégé owl plugin: An open development environment for semantic web applications," *The Semantic Web*, pp. 229–243, 2004.

[21] D. McGuinness, F. Van Harmelen *et al.*, *OWL web ontology language overview*, http://www.w3.org/TR/owl-features/, 2004.

[22] V. Mascardi, V. Cordi, and P. Rosso, *A Comparison of Upper Ontologies*, 2006, http://www.disi.unige.it/person/MascardiV/Download/DISI-TR-06-21.pdf.

[23] M. J. OConnor and A. K. Das, "A method for representing and querying temporal information in owl," *Biomedical Engineering Systems and Technologies, Communications in Computer and Information Science*, pp. 97–110, 2011.

[24] C. Fellbaum, "Wordnet," *Theory and Applications of Ontology: Computer Applications*, pp. 231–243, 2010.

[25] I. Niles and A. Pease, "Towards a standard upper ontology," in *International Conference on Formal Ontology in Information Systems*, 2001, pp. 2–9, http://www.ontologyportal.org/.

[26] P. Barnaghi, M. Compton, O. Corcho, R. G. Castro, J. Graybeal, A. Herzog, K. Janowicz, H. Neuhaus, A. Nikolov, and K. Page, *Semantic Sensor Network XG Final Report*, 2011, http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/.

[27] P. Kostelnik, M. Sarnovsky, J. Hreno, M. Ahlsen, P. Rosengren, P. Kool, and M. Axling, "Semantic devices for ambient environment middleware," *EURO TrustAmi*, pp. 18–19, 2008.

[28] M. Vallée, F. Ramparany, and L. Vercouter, "Dynamic service composition in ambient intelligence environments: a multi-agent approach," in *1st European Young Researcher Workshop on Service-Oriented Computing*. Citeseer, 2005, pp. 1–6.

[29] J. Fan, K. Barker, B. Porter, and P. Clark, "Representing roles and purpose," in *International Conference on Knowledge Capture*, British Columbia, Canada, October 22–23 2001, pp. 38–43.

[30] E. Sunagawa, K. Kozaki, Y. Kitamura, and R. Mizoguchi, "Organizing role-concepts in ontology development environment: Hozo," *AI Technical Report*, vol. 4, pp. 453–468, 2004.

[31] F. Steimann, "On the representation of roles in object-oriented and conceptual modeling," *Data and Knowledge Engineering*, vol. 35, no. 1, pp. 83–106, 2000.

[32] J. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. New York, USA: Addison Wesley Publishing Company, 1983.

[33] S. Moralee, M. Uschold, M. King, and Y. Zorgios, "The enterprise ontology," *The Knowledge Engineering Review*, vol. 13, 1998.

[34] A. Hristoskova, W. Boffé, T. Tourwé, and F. De Turck, "Integrated semantic and event-based reasoning for emergency response applications," in *8th International Conference on Geo-information for Disaster*, 2012.