Modular Semantic CEP for Threat Detection

Karl Hammar

Jönköping University, Jönköping, Sweden
haka@jth.hj.se

Abstract. This paper introduces a generic architecture for semantic complex event processing (CEP) over sensor data, as exemplified by a reference implementation system in development for the security and surveillance domain. The system gathers data from a number of sensor subsystems and classifies this data according to ontology-based situation models and rules. The output of the system is alerts about threat situations that supports human operators in deciding when and how to deploy security personnel to manage these threats. The novelty of the proposed approach lies in the use of modular ontology design patterns for system configuration, which enable non-technical users to rapidly configure the system for particular scenarios.[1]

1 Introduction

Complex Event Processing (CEP) is a set of methods for scanning through time-indexed data in order to detect patterns signifying the presence of particular events or situations that are of interest. Possible usages of such processing exist in a variety of areas, from improving operational efficiency in healthcare [12] to analysis of trending topics in social networks [3].

While much CEP work has been performed using forward-chaining rules and heuristics, it has recently been found that adding semantic technologies to CEP provides a number of benefits, including the ability to detect more complex situations and detect situations using background knowledge [1]. Using semantic technologies also enables the reuse of both existing semantic models, and established research on knowledge modeling, knowledge management, and reasoning using knowledge bases.

A CEP system deployed in the mobile security and surveillance domain would need to both be able to detect threatening situations, and to be easily reconfigured for different scenarios (as threats at sports events are of a different nature than those at political summits, for instance). Aiming to meet these needs, the author has developed an architecture for performing semantic CEP in which system configuration is performed by way of ontology design patterns (ODPs). Such patterns, small reusable pieces of ontologies, have previously proven helpful in allowing non-technical users to construct semantic models [5]. By pre-configuring particular system behaviors and entity definitions and encoding them into pluggable ODP pattern modules, one can make reconfiguration and redeployment of the system as easy as selecting the modules relevant for the deployment scenario, and inputting the specific conditions existing at the particular site of deployment. These tasks could be performed by a domain expert without extensive knowledge engineering training.

While the implementation of the proposed architecture into a working system is still in a prototype stage, it will within the coming months be deployed and benchmarked in a number of scenarios against a non-semantically enhanced alternative within a large-scale EU project.

[1] The research presented in this paper was supported by the Swedish Foundation for Internationalisation in Higher Education and Research, project "Development and Evolution of Ontologies"

2 Related work

2.1 Complex Event Processing

Complex Event Processing is introduced by Luckham & Frasca in [9]. In their approach, patterns based on temporal or causal links between events are defined and formalized into mapping rules. When executed over incoming time-indexed data streams, pattern connect lower level basic events to form higher level complex events. The approach is exemplified by a factory scenario, where the events concern communication with automated production machinery and mappings can be made between low-level network communication events and higher level workflow events such as *"begin process"* or *"setup machine"*. Luckham develops these ideas further in [8], where he describes how similar approaches can be applied to message-oriented information systems in a variety of domains.

CEP based on sensor data feeds has been explored in many papers, often using RFID sensors. [11] illustrates how CEP over RFID data can provide advantages in retail management and healthcare, and develop an event description language appropriate for such data streams. [10] uses cameras and accelerometers as input to detect harmful situations in home care for the elderly. [12] describes a CEP system to improve patient safety and operational efficiency in an RFID-enabled hospital.

As indicated by [1] current CEP approaches however have some drawbacks, particularly in terms of recognizing events using background knowledge. Only those relations between events and entities which are made explicit in the input data stream can be used for detection and correlation purposes. The author would to this add two further drawbacks of current rule-based CEP systems. The rules used in such systems are expressed in formal languages such that extensive knowledge of logics and knowledge engineering is required to configure a system for a particular CEP task. They are simply not very user-friendly. Furthermore, since the emphasis in such systems tend to be on rules rather than the events and detected entities that are used for CEP (their ontologies, i.e. the event hierarchies used for event classification, are often rather small), reusability of event and entity definitions from other information systems is limited.

2.2 Semantic Complex Event Processing

In order to overcome some of the limitations of pure rules-based CEP systems, [1] suggests the use of Semantic Complex Event Processeing (SCEP), in which background knowledge is encoded into knowledge bases that are accessed by a rules engine to support CEP. Apart from enabling reasoning over domain and background knowledge, this also enables detection of more complex situations, recommendations, event classification, clustering, filtering, etc.

Another approach to enabling semantic processing of time-indexed data is proposed by Barbieri et al. in [4] and further developed in [3] and [2]. Their contribution is twofold – to begin with they propose an extension of the SPARQL query language commonly used to query knowledge bases, enabling continuous querying over timestamped RDF graphs using configurable sliding windows. They also develop support for reasoning over such such sliding windows, including dropping both facts and inferred knowledge once the window has passed (greatly reducing the computational power required to calculate inferences). Based on these approaches it is possible to construct SCEP systems using only semantic technologies.

2.3 Ontology Design Patterns

The knowledge bases used in SCEP systems are defined by ontology schemas. These ontologies are models that define what concepts are available to reason about, and what vocabularies are used

to define the relations between concepts. Ontology engineering is the art or science of developing ontologies. Much effort has been put into the development of ontology engineering tools, techniques, and methods. However, it is still a time-consuming human engineering process that cannot be fully automated or scripted, since it requires engineering decisions and design skills.

Saving time in ontology engineering by reuse of best practices in the form of *ontology patterns* or *ontology design patterns* (the terms are sometimes used interchangeably) have been proposed independently by Blomqvist and Sandkuhl [6], and Gangemi [7]. Such patterns, encodings of best practices into small reusable blocks, are intended to reduce the need of extensive experience when developing ontologies and thus speed up development. The latter work's perspective on ontology design patterns has become widespread, and a community has formed[2] based on the further developments of these ideas presented in the NeOn project.

3 Proposed architecture

The core of the system is a live observation knowledge base, defined according to an ontology schema. The ontology consists of both general features that are always relevant in the context of such a system (vocabularies of time, geographical locations and distances, sensor metadata, etc), and of features that are scenario and deployment specific. The latter features are imported from a set of four configuration knowledge bases, that together define system behavior. These knowledge bases define, respectively: scenario configuration (i.e. background/context knowledge), situation correlation behavior, observation/entity fusion behavior, and critical situation detection behavior. Their ontology schemas are constructed by importing patterns from an ontology design pattern repository.

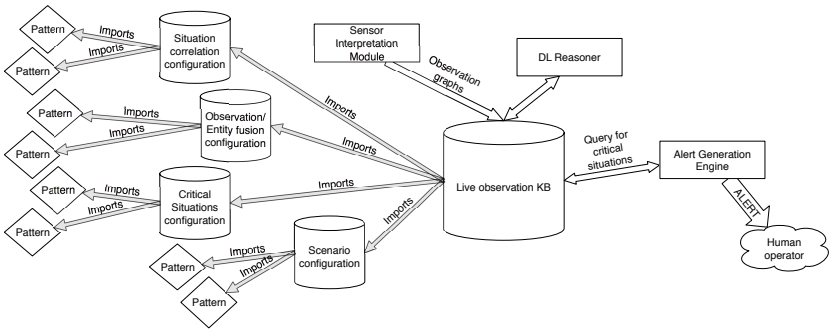

Fig. 1. Semantic Knowledge Fusion system architecture.

Data input from the deployed sensor subsystems is mapped to the general ontology vocabulary by sensor interpretation modules and stored as observation graphs in the knowledgebase. A description logic (DL) stream reasoner (as proposed by [2]) is executed on the knowledgebase and inferences about possible threats are made. The system architecture is illustrated in Figure 1.

[2] http://www.ontologydesignpatterns.org

3.1 Modeling and Handling Situations

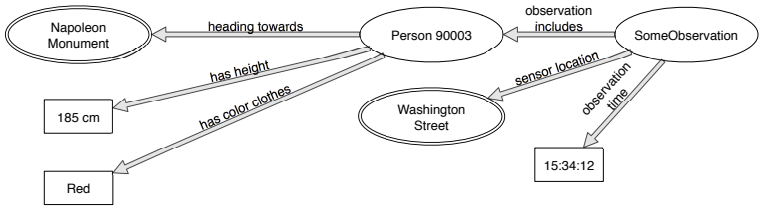

Fig. 2. Observation Graph example. Data properties are illustrated by squares, preconfigured entities by double-lined ellipses, and newly detected entities by single-line ellipses.

In the system, sensor observations are represented as RDFS graphs, in which an observation is linked via properties defined in the general ontology or scenario configuration ontology to entities (expressed as RDFS instances) and metrics (expressed as RDFS literals) that describe it. Entities linked to an observation in this manner can be either new entities also observed via sensors, or entities that are predefined in the scenario configuration knowledge base. See Figure 2 for an illustration of the concept.

Prototypical critical situation models are defined in the critical situation configuration knowledge base using description logic axioms, and any observations matching these situation models are automatically by the DL reasoner inferred to be observations of such threatening situations. For instance, a threatening situation could be (expressed in human terms) "large noisy crowd near conference hotel". When observation graphs in the live observations knowledge base exist that correspond to this particular situation model, a threat observation is issued, in turn raising an alert to a human operator.

Observation/Entity Fusion An important requirement on a SCEP system is the capability to determine whether two observation graphs are in fact observations of the same situation, and whether two observed entities from different observation graphs are in fact the same thing. Such matching capability is in the prototype developed using Jena Rules and placed within the Ontology/entity fusion knowledge base.

Observation Correlation Another requirement on the system is that it should be able to detect which observed situations are potentially interesting only when co-occurring. Such artificial creation of composite threat observations is not possible using purely description logic languages, as these only allow inferencing based on existing knowledge and not the creation of entirely new facts. However, it can be accomplished using ontology-based rule languages (such as Jena Rules) that have built-in instance generation.

3.2 System Configuration

The advantage of using patterns is twofold – firstly, it enables quicker reconfiguration of the system, and secondly, it lowers the barrier of entry in terms of knowledge engineering experience needed to

configure the system. Using patterns to configure the system is a two-step process. First, contextual or background knowledge is imported and configured into the scenario configuration knowledge base, and then a set of threat detection patterns are imported into the three knowledge bases governing system behavior. The use of the Web Ontology Language (OWL) enables a pattern with a given URI to be imported via the addition of a single statement to the ontology, so no reengineering is required – it is simply plug and play.

Scenario Configuration To configure the system for a particular deployment scenario, a vocabulary of the type of entities and relations expected within this scenario are required to be imported to the scenario configuration knowledge base ontology. These patterns build upon the general features present in the system to begin with, and specialize them as needed. For instance, the inclusion of a "Metro system" pattern subclasses concepts such as "Geographical place", "Time interval", and "Path" to enable the modeling of a number of stations, a train schedule, and the metro lines that pass these stations. At the time of writing no purpose-built GUI exists for this task, but it is not farfetched to envisage such a GUI based on simple drag and drop metaphors.

After configuring the scenario configuration knowledge base vocabulary in this manner, facts about the actual deployment scenario are added to it. This involves entering the actual metro stations, train schedules, distances between them, and so on. Some of this data could probably be automatically imported from existing information systems (though the interface requirements for this have not yet been studied), and some of it be input by deployment staff using for instance forms-based interfaces.

Threat Detection Configuration Once the scenario configuration is in place, patterns can be imported into the three system behavior knowledge bases (i.e. critical situation detection, observation/entity fusion, and situation correlation). These patterns can be expressed either using pure description logics, or using rule languages for the semantic web. Either way, they will be packaged as small ontology components, and can easily be imported, again using a drag-and-drop interface.

4 Example Scenario

In order to exemplify how the prototype system based on the proposed architecture can be configured to support threat detection, a hypothetical scenario has been developed in cooperation with domain experts. The scenario concerns an international political summit on the environment, taking place at a convention facility in Copenhagen, Denmark. At and around the facility a surveillance perimeter is set up with a number of sensor subsystems, including intelligent camera systems (capable of face detection and object tracking), audial surveillance, seismic sensors, terahertz imaging, etc. In the following sections some threats applicable in this scenario are described, and the corresponding models for threat detection explained.

4.1 Bombs at Bridges

VIP motorcades will travel on defined routes from the downtown hotels where the VIPs are accommodated to the convention center. Along the way, they will pass a number of bridges across waterways. A possible attack vector is for terrorists to plant bombs at these bridges, in order both to

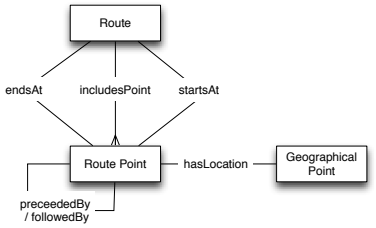

Fig. 3. Route ontology design pattern

```
SELECT ?route ?location
WHERE {
  ?route rdf:type :Route .
  ?route :routeIncludesPoint ?routepoint .
  ?routepoint :routePointHasLocation ?location .
  ?location rdf:type :Bridge
}
```

Fig. 4. SPARQL query returning bridges passed by motorcades.

damage the motorcades, and to cut off access for paramedics and security personnel. It is therefore important to detect when packages are placed at several of these bridges at the same time.

In this scenario, during pre-deployment configuration the Route Ontology Design Pattern (Figure 3) is imported into the scenario configuration knowledge base, and the knowledge base is then loaded with information about the routes that the motorcades are ordered to take, expressed according to the pattern vocabulary. This enables querying the knowledge base for bridges that motorcades will pass (Figure 4).

```
(?pab1 rdf:type :PackageAtKeyBridge) (?pab2 rdf:type :PackageAtKeyBridge)
(?pab1 :observationConcernsRoute ?route) (?pab2 :observationConcernsRoute ?route)
makeTemp(?to) ->
(?to rdf:type :Threat) (?to :observationConcernsRoute ?route)
```

Fig. 5. Rule for detecting multiple packages at bridges.

Additionally a threat detection pattern consisting of two rules is imported into the situation correlation knowledgebase. At runtime, these rules identify and correlate incoming observations of unknown packages at motorcade-passing bridges, and generate a threat observation when more than one of the bridges on a particular route are involved in such package observations (Figure 5). The threat alert generation engine will based on this threat observation bring up an alert to a human operator and allow him/her to deploy preventative measures such as changing the route or calling in a bomb squad.

4.2 Crowd Management

It is expected that crowds of protesters will gather near the conference venue. Some of these crowds will contain hooligans that need special monitoring, especially as they approach high-value locations such as the main conference center or VIP hotels. Hooligan behavior can be detected via acoustic sensors (for instance, the noise of crushed glass), via video surveillance (rapid crowd movements, protesters wearing facial covering), via seismic detection of unusual crowd movement patterns, etc. This detection is performed via traditional CEP methods, using simple rules and heuristics.

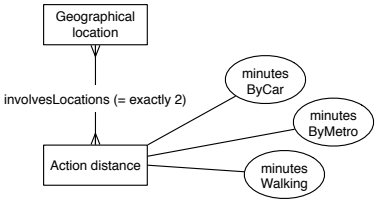

Fig. 6. ActionDistance ODP

```
(:HooliganObservation1 :observationAtPlace ?hloc)
(?ad ActionDistance:distanceInvolvesLocation ?hloc)
(?ad ActionDistance:distanceInvolvesLocation ?tloc)
(?tloc rdf:type :HighValueLocation)
(?ad ActionDistance:distanceInMinutesByCar ?d)
greaterThan(?d, 5) makeTemp(?to)
-> (?to rdf:type :Threat)
(?to :threatInvolves :HooliganObservation1)
(?to :threatInvolves ?tloc)]
```

Fig. 7. Threat detection pattern detecting hooligans close to high-value targets

In order to respond to hooliganism and classify the threat levels of different hooligan situations, the ability to reason about the distance of such situation to high-value locations and the required response time of police forces is important, and in order to do so, background knowledge as provided by semantic CEP is required. To represent these response times, the ActionDistance ODP (Figure 6) is imported to the scenario configuration knowledgebase during pre-deployment, and relevant distances between points of interest entered according to the semantics of this pattern. A pluggable threat detection pattern is then imported to critical situation configuration knowledgebase, such that at runtime an alert will be raised if any hooligan behavior is detected nearer to a high-value location than a police force is able to respond. See Figure 7 for such a pattern. Though the pattern is here simplified to assume police response time to any locality is always five minutes, it could easily be expanded to model arbitrary response times based on current location of police forces.

4.3 Unauthorized Access

In the described scenario, a key requirement is the ability to trigger an alarm when an unauthorized person is detected within the conference facilities. The deployed camera system has the ability to identify individual persons by using face recognition techniques, but in order to know whether detected people are allowed access to certain areas higher level reasoning is needed.

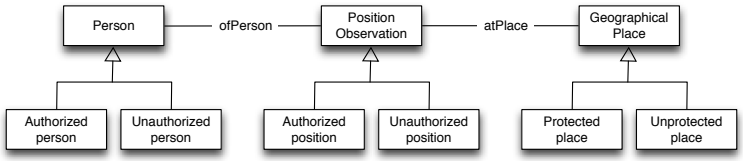

Fig. 8. AuthorizedPosition ODP.

In order to support such reasoning, the AuthorizedPosition ODP (Figure 8) is imported into the scenario configuration knowledgebase. Known VIPs and known protected areas are input into

the same knowledgebase. Finally, a threat detection pattern is imported to the critical situation recognition knowledgebase that triggers a threat observation and operator alert upon observations of unauthorized persons at protected places.

5 Conclusions and Future Work

The conceptual work and prototyping thus far indicates that the proposed architecture is a viable way toward constructing a rapidly reconfigurable system for knowledge fusion and complex event processing in the security and surveillance domain. The next steps in this work are to do benchmarking of the prototype system against a system built using traditional CEP techniques, in order to compare performance of the two solutions, and to evaluate whether the expressivity of the used ODPs are sufficient.

The use of Ontology Design Patterns has previously been shown to help in constructing ontologies, but the use of such patterns as configuration modules has to the author's knowledge not been tried before. The author aims to study this use of patterns further, in particular the perceived usability of packaging patterns and rules as reusable modules, how such modules are to be structured and documented, and what type of user interfaces are required to simplify the reuse of them.

References

1. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in etalis. Semantic Web (2011)
2. Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. The Semantic Web: Research and Applications pp. 1–15 (2010)
3. Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Huang, Y., Tresp, V., Rettinger, A., Wermser, H.: Deductive and inductive stream reasoning for semantic social media analytics. Intelligent Systems, IEEE PP: 99, 1–1 (2010)
4. Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-sparql: Sparql for continuous querying. In: Proceedings of the 18th international conference on World wide web. pp. 1061–1062. ACM (2009)
5. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on pattern-based ontology design. In: Proceedings of the fifth international conference on Knowledge capture. pp. 41–48. ACM (2009)
6. Blomqvist, E., Sandkuhl, K.: Patterns in ontology engineering: Classification of ontology patterns. ICEIS (3) pp. 413–416 (2005)
7. Gangemi, A.: Ontology design patterns for semantic web content. The Semantic Web–ISWC 2005 pp. 262–276 (2005)
8. Luckham, D.: The power of events: an introduction to complex event processing in distributed enterprise systems. Addison-Wesley Longman Publishing Co., Inc. (2001)
9. Luckham, D., Frasca, B.: Complex event processing in distributed systems. Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford (1998)
10. Søberg, J., Goebel, V., Plagemann, T.: Commonsens: Personalisation of complex event processing in automated homecare. In: Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2010 Sixth International Conference on. pp. 275–280. IEEE (2010)
11. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. pp. 407–418. ACM (2006)
12. Yao, W., Chu, C., Li, Z.: Leveraging complex event processing for smart hospitals using rfid. Journal of Network and Computer Applications (2010)