Communications in Mobile Computing
a SpringerOpen Journal

ORIGINAL SUBMISSION Open Access

User-centric modeling and processing for ubiquitous events using semantic capability models

Feng Gao[1*], Sami Bhiri[1] and Zhangbing Zhou[2,3]

*Correspondence: feng.gao@deri.org
[1]Digital Enterprise Research Institute, National University of Ireland, Galway, Galway, Ireland
Full list of author information is available at the end of the article

Abstract

Recent development in sensor networks and mobile computing is gaining increasing interest from enterprises. Sensor data services can provide fine-grained information of the physical world and feed the event processing systems where high level business logic is evaluated and coarse-grained business events are derived. However current event processing systems are not user-oriented and take considerable effort to configure and implement mainly because of the granularity mismatch. In this paper, we present a intuitive way to model the complex event patterns based on semantic descriptions of sensor service capabilities. Then, we transform the event patterns into stream queries to facilitate an automatic implementation for the event processing system.

Introduction

Business process management (BPM) tools, including business process modeling environments, process engines and process analyzing tools, can be used to design, implement, execute and reengineer business processes. Today enterprises are demanding more flexibility from BPM to adapt the fast changing business environment in-time and costlessly. On the other hand, development in sensor networks is gaining increasing interest from enterprises. Many efforts have been made to integrate sensor functionalities with enterprise systems to manage business processes more dynamically. A natural use of sensors is to monitor the state changes of the real-world and trigger event driven processes or actions, thus complex event processing (CEP) techniques are crucial in the context of sensor-aware process management. Most current CEP tasks are delegated to standalone CEP engines. These engines are usually equipped with rule-based engines to define and analyze complex event patterns, and require some programming skills to encapsulate the corresponding event data to communicate with rule engines. Unfortunately neither rule languages or programming APIs are friendly enough for business users. As such, companies need significant technical efforts to implement a business process involved with CEP.

To accelerate the implementation for ubiquitous event processing in enterprise systems and increase the flexibility of process management, we present a user-centric way to model complex events with graphical notations. Then, we develop algorithms to transform event patterns into stream queries which can be executed directly by stream reasoning engines. We also propose a semantic model to describe sensor capabilities, we believe the capability model can help business users to select the primitive events they need while composing complex event patterns.

Related works

Stream Reasoning is an emerging research area that tries to enable reasoning on continuous data and support processing for both dynamic data and static background knowledge. In [1] a prolog-based framework is proposed to transform continuous triples as logic facts and stream query as rules, so that the processing of complex events can make use of both dynamic event data and static background knowledge. In [2] the authors use a "white box" approach and support native stream reasoning operators, they also provide means to optimize the query plan so that the evaluation of stream query can be more efficient. We will transform our user-centric complex event definition into the executable stream query language defined in [2] (with possible extensions) to enable automated implementation of complex event patterns.

Semantic CEP is discussed in several works. In [3] the authors elaborates the benefits to extend syntactical event correlation to semantic event correlations. In [4] ontology is used to describe event rules as well as context-aware devices (sensors), an event hierarchy is used to model the causal relationship between different levels of events. In the framework propsed in [5], users can select and correlate sensors based on the semantic sensor description, then, a semantic middleware will translate the users' requirements into the internal language used by the CEP engine (e.g. EPL) and program the sensors to prepare the streams. Very few of the work above provide a formalization of the event pattern language they use. Moreover, they are not friendly enough for business users and can only operate on predefined event streams.

Graphical event notations and stream queries

BEMN [6] intend to provide a graphical representation for the event composition languages beyond conventional textual language. BEMN is able to describe the business event patterns identified in [7]. An example of the complex event pattern in the BEMN is shown in Figure 1.

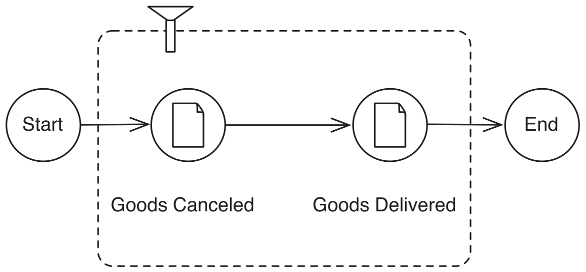

Figure 1 Example inhibition event pattern. The event pattern demonstrates the following scenario: RFID tags of the goods are scanned while they arrive, if goods have been delivered which were previously canceled, a delivery exception event is raised.

Despite that the formal semantics of the language are defined and the execution environments are described, the original language did not take into account how stream processing technique can be integrated to enable the execution of event models. As a result, some simplified matching functions are defined and based on which a more restrained core composition model can be directly executed by a single transactional matching. This has limited the expressiveness of executable event composition models and introduce the overhead of translating general (non-core) models into core models.

Furthermore, the original execution environment exposes all the events to all the subscription scopes (process engine, process instances or activities) through a single event channel. This will greatly impact the efficiency of the matching functions when the system scales.

Moreover, the original BEMN language does not specify the data structure of event declarations. It is left to the programmers who implement the event rules. In this way, the technical details are hidden from the business users, but the level of execution automation is compromised.

We propose to describe sensor capabilities with our semantic model to help business users discover the primitive events they need, as well as create filters upon event data. To overcome the above limitations, we revise the BEMN language and develop an algorithm based on *Program Structure Tree* [8] to transform BEMN patterns into stream queries.

Sensor service capability description

Service Oriented Architecture (SOA) decouples service consumers from providers and provide means to build distributed programs with more flexibility. A service capability is about what a service does. The concept of service capability plays a central role in service description. Various efforts have been made to match web services based on service description [9-11] or service protocols [12]. In this paper we propose to help business users to identify adequate web services that deliver primitive events by exploring the service capabilities.

Service capability meta-model

We use the capability meta-model described in [13] to create a capability model for sensor services using semantic data.The basic ontology for meta-model is very simple and intuitive. A capability is modeled with a set of attributes and attribute-values. We use the human-readable notation-3 format for the semantic representation throughout the paper, as shown in Table 1.

Hierarchical capability models can be constructed to model capabilities on different abstraction levels by using the relationships we define between capabilities. In particular, we define *specify* and *extend* relationship between 2 capabilities, both relationships represent a refining process for the capabilities. Using these refining relationships, we can reuse

Table 1 Snippet of capability meta-model

:Capability	a	rdfs:Class, owl:Class.
:Attribute	rdfs:subClassOf	rdfs:Property;
	rdf:domain	[owl:unionOf :AttributeValue, :Capbility];
	rdf:range	:AttributeValue.
:AttributeValue	owl:equal	rdfs:Resource.

abstract capabilities models to create more concrete ones. For 2 capabilities C_1 and C_2: C_1 *specify* C_2 *iff* C_1 and C_2 has the same set of attribute but C_1 has more concrete attribute values; C_1 *extend* C_2 *iff* C_1 has more attributes than C_1 and their shared attributes has the same values. Notice that in the above core meta-model definitions, both capability and attribute-value can use attributes to describe them.

Dynamicity of attribute values can be modeled in our capability meta-model with constraints or data-fetching endpoints. There are various reasons to have dynamic attribute values in the service description, e.g.: attribute inter-dependency, inherent dynamicity and etc. Modeling the dynamicity empowers us to derive *capability offers* at run-time with information unavailable at design time. We leverage the datafetching technique described in [14] to retrieve dynamic information and create service offers. We distinguish between *conditional values* and *dynamic values*. Conditions and constraints are expressed with SPARQL query segments and dynamic values will give an service endpoint as an access point for datafetching. During service discovery phase the discovery engine will use the SPARQL segments and URIs to construct semantic descriptions for the service offers and match them against search requests.

Sensor service capability ontology

Using the capability meta-model described above, we are able to create an ontology for sensor capabilities. Our ontology reuses some terms and relations from the comprehensive SSN ontology[a]. Core classes and properties for the sensor capability ontology is shown in Figure 2. In the sensor capability ontology the namespace “*cap:*" refers to the capability meta-model ontology, “*ssn:*" refers to the SSN ontology, “*sc:*" is the target namespaces, which is the sensor capability ontology.

As shown in the UML diagram, the *sc:SensorCapability* is a sub-class of capability. Five attributes (sub-properties of *cap:Attribute*) are used to describe a sensor capability. A top level sensor capability as the root in the sensor capability hierarchy is defined in Table 2.

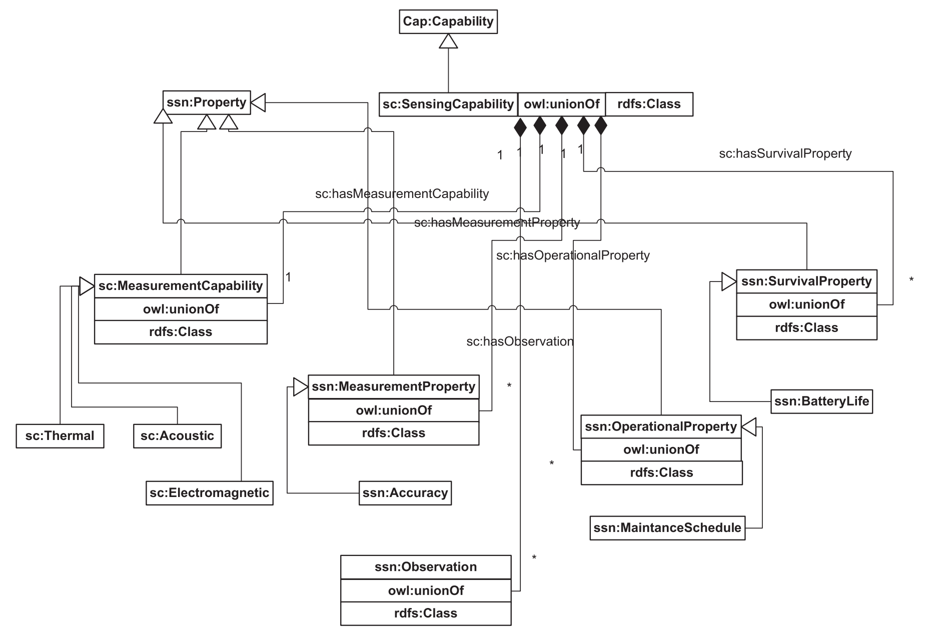

Figure 2 Sensor capability ontology.

Table 2 Snippet of top level sensor capability

sc:TopSensorCapability	a	sc:SensorCapability;
	sc:hasMeasurementCapability	sc:MeasurementCapability;
	sc:hasMeasurementProperty	ssn:MeasurementProperty;
	sc:hasOperationalProperty	ssn:OperationalProperty;
	sc:hasSurvivalProperty	ssn:SurvivalProperty;
	sc:hasObservation	ssn:Observation.

Note that the domains and ranges of these attributes are unioned with *rdfs:Class* to allow both instances and sub-classes as domains and ranges so that we can identify the refining relations we previously defined.

The measurement capability is similar to the original *ssn:MeasurementCapability* which describes the functional properties of a sensor, i.e what does a sensor observes. In SSN ontology there is a one-to-one mapping between instances of measurement capability and measurement property, and these measurement properties can vary depending on environmental conditions. For example, the accuracy of a temperature sensor can be influenced by air temperature. To model the conditional values the SSN ontology uses ranged values with upper and lower bounds to create several divisions of the air temperature, and create multiple instances of the accuracy property. Then, each accuracy instance is assigned to a temperature division. However no general approach is described in SSN ontology for modeling more complex conditions nor for more dynamic properties which are not constrained by static rules and requires data-fetching. Such need can be easily fulfilled by using the mechanism we introduce in the capability meta-model. An example of describing the dynamic accuracy for a temperature sensor is shown in Table 3.

In the above example, a temperature sensor is modeled as a variant of the top sensor capability by extending and specifying some attributes. The observation attribute value of a sensor is, and always will be, a dynamic value. A service endpoint is specified for performing the datafetching task and a lifting schema is used for data transformation.

Table 3 Snippet of a specific sensor capability

sc:Sensor1	a	ssn:SensingDevice,sc:SensorCapability;
	sc:hasMeasurementCapability	sc:Thermal;
	sc:hasMeasurementProperty	sc:Mp1;
	sc:hasOperatingProperty	sc:Op1;
	sc:hasSurvivalProperty	sc:Sp1;
	sc:hasObservation	[rdfs:subClassOf ssn:Observation,cap:DynamicValue;
		sc:hasEndpoint "www.deri.org/sensor1"^^xsd:URI;
		sc:hasLifting "www.deri.org/lifting.xsd"^^xsd:URI].
sc:Mp1	a	sc:Accuracy, cap:ConditionalValue;
	cap:hasRequired	[cap:hasAttribute sc:hasObservation;
		cap:asVariable "?temp"].
	cap:hasCondition	[cap:hasValue "0.8"^^xsd:decimal;
		sc:hasExpression "Filter(?temp <10 && ?temp >0)"^^
		cap:SPARQL];
		[cap:hasValue "0.9" xsd:decimal;
		sc:hasExpression "Filter(?temp >10)"^^cap:SPARQL].
sc:Sp1	a	ssn:BatteryLife;
	cap:hasValue	"1"^^xsd:decimal;
	cap:hasUnit	dbpedia:Month.

The accuracy of the measurement is defined as a conditional value. The accuracy has a required parameter, which is the air temperature measured by the sensor. Note that multiple conditions can be used to assign different values to a condition attribute value, but these conditions must be disjoint. When the discovery engine encounters a query on the accuracy, it will recognize the accuracy as a conditional value and check if there's any required attributes for evaluating the condition. If the condition relies on a dynamic value, the engine will perform the datafetching task to retrieve the concrete value and formulate a query for the conditional value. Multiple queries will be created and evaluated for multiple conditions. There will be at most 1 positive result since all conditions are disjoint. A sample query for the accuracy is shown below.

```
SELECT ?acc
WHERE {_x a ssn:Observation;
         cap:hasValue ?temp BIND(0.8, ?acc )
FILTER(?temp<10 && ?temp>0)}
```

Conclusions and future work

In this paper we present a novel approach to facilitate user-centric modeling and automated implementation for complex event patterns based on ubiquitous data service. We revise the BEMN language to allow more expressive and executable models as well as more specific descriptions on primitive event declarations. Then we translate the event patterns defined by the business users to a stream reasoning query so that they can be evaluated immediately without further coding. We also provide a semantic model to describe sensor service capabilities on different abstraction levels and while resolving the dynamicity in service descriptions.

As a future work we intend to implement and evaluate the system. We also intend to support interval events to model complex events and introduce temporal relations between interval events. An efficient discovery and navigation mechanism for the hierarchy of capabilities is also on the agenda.

Endnote

[a]http://www.w3.org/2005/Incubator/ssn/wiki/Semantic_Sensor_Net_Ontology

Acknowledgements
This work is supported by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), and is partially supported by the Fundamental Research Funds for the Central Universities (China University of Geosciences at Beijing).

Author details
[1]Digital Enterprise Research Institute, National University of Ireland, Galway, Galway, Ireland. [2]School of Information Engineering, China University of Geosciences (Beijing), Beijing, China. [3]Computer Science Department, TELECOM SudParis, Paris, France.

Received: 9 August 2012 Accepted: 13 August 2012
Published: 2 October 2012

References
1. Anicic D, Fodor P, Rudolph S, Stojanovic N (2011) EP-SPARQL: a unified language for event processing and stream reasoning. In: Proceedings of the 20th international conference on World wide web, WWW '11, New York, NY, USA, ACM, pp 635–644
2. Le-Phuoc D, Dao-Tran M, Parreira JX, Hauswirth M (2011) A native and adaptive approach for unified processing of linked streams and linked data. In: Proceedings of the 10th international conference on The semantic web - Volume Part I, ISWC'11, Berlin, Heidelberg, Springer-Verlag, pp 370–388
3. Moser T, Roth H, Rozsnyai S, Mordinyi R, Biffl S (2009) Semantic Event Correlation Using Ontologies. In: Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part II, OTM '09, Berlin, Heidelberg, Springer-Verlag, pp 1087–1094

4. Li Z, Chu CH, Yao W, Behr RA (2010) Ontology-Driven Event Detection and Indexing in Smart Spaces. In: Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing, ICSC '10, Washington, DC, USA, IEEE Computer Society, pp 285–292
5. Taylor K, Leidinger L (2011) Ontology-driven complex event processing in heterogeneous sensor networks. In: Proceedings of the 8th extended semantic web conference on The semanic web: research and applications - Volume Part II, ESWC'11, Berlin, Heidelberg, Springer-Verlag, pp 285–299
6. Decker G, Grosskopf A, Barros A (2007) A Graphical Notation for Modeling Complex Events in Business Processes. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, EDOC '07, Washington, DC, USA, IEEE Computer Society, p 27
7. Barros A, Decker G, Grosskopf A (2007) Complex events in business processes. In: Proceedings of the 10th international conference on Business information systems, BIS'07, Berlin, Heidelberg, Springer-Verlag, pp 29–40
8. Johnson R, Pearson D, Pingali K (1994) The program structure tree: computing control regions in linear time. SIGPLAN Not 29(6): 171–185
9. Kopecký J, Vitvar T, Bournez C, Farrell J (2007) SAWSDL: Semantic Annotations for WSDL and XML Schema. IEEE Internet Computing 11(6): 60–67
10. Martin D, Burstein M, McDermott D, McIlraith S, Paolucci M, Sycara K, McGuinness DL, Sirin E, Srinivasan N (2007) Bringing Semantics to Web Services with OWL-S. World Wide Web 10(3): 243–277
11. Zaremba M, Vitvar T (2008) WSMX: a solution for B2B mediation and discovery scenarios. In: Proceedings of the 5th European semantic web conference on The semantic web: research and applications, ESWC'08, Berlin, Heidelberg, Springer-Verla, pp 884–889
12. Zhou Z, Bhiri S, Zhuge H, Gaaloul W (2012) Assessment of Service Protocol Adaptability Based on Novel Walk Computation. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on PP(99):1–32
13. Bhiri S, Derguech W, Zaremba M (2012) Web Service Capability Meta Model. In: Krempels KH, Cordeiro J (eds) WEBIST, SciTePress, pp 47–57
14. Zaremba M, Vitvar T, Bhiri S, Hauswirth M (2011) Service Offer Discovery Using Genetic Algorithms. In: Proceedings of the 2011 IEEE Ninth European Conference on Web Services, ECOWS '11, Washington, DC, USA, IEEE Computer Society, pp 23–30

doi:10.1186/2192-1121-1-7
Cite this article as: Gao *et al.*: User-centric modeling and processing for ubiquitous events using semantic capability models. *Communications in Mobile Computing* 2012 **1**:7.