Context Management in Event Marketplaces

Yiannis Verginadis[1], Ioannis Patiniotakis[1], Nikos Papageorgiou[1], Dimitris Apostolou[1], Gregoris Mentzas[1], Nenad Stojanovic[2]
[1] Institute of Communications and Computer Systems,
National Technical University of Athens,
{jverg, ipatini, npapag, dapost, gmentzas}@mail.ntua.gr
[2] FZI Forschungszentrum Informatik, Karlsruhe, Germany
nstojano@fzi.de

Abstract. This paper refers to methods and tools for enabling context detection and management based on events. We propose a context model that builds on top of previous efforts and we give details about the mechanisms developed for context detection in event marketplaces. In addition, we show how simple or complex events can be used in combination with external services in order to derive higher level context with the use of Situation-Action-Networks (SANs). Specifically, we present two different approaches, one for detecting low level context and another one for deriving higher-level contextual information using SANs. We present an illustrative scenario for demonstrating the process of specialization of our generic context model and its instantiation based on real-time events.

Keywords: Context, Event Marketplace, Detecting Context, Deriving Context

1 Introduction

Context is “any information that can be used to characterize the situation of an entity, i.e., a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” [1]. Context detection is considered important in the so-called event marketplaces [2] (see e.g., http://pachube.com, a platform offering a service based architecture, a range of graphing and visualization tools, event detection via triggers, along with cost-effective data storage) for enhancing the user's experience when interacting with the event marketplace.

Events from event marketplaces are an important source of context for service-based applications that consume them because they may convey important information, which is relevant for service execution and adaptation. To achieve the goal of injecting event processing results to context, an event-based context model is needed along with context detection and derivation mechanisms. In previous work [3] Situation-Action-Networks have been proposed as a hierarchical goal-directed modeling approach comprising nodes with specific semantics used to model goal decompositions, enriched with flow control capabilities. SANs provide means to decompose goals into subgoals and capabilities for seeking and achieving the high-

level goals, involving situations (i.e. complex event patterns), context conditions and actions.

The simplest SAN possible is a two level tree with a parent (root) node and three child nodes, each of them having specific semantics. A parent node models the Goal sought. The leftmost child node describes a situation that must occur, in order to start goal seeking. The middle child node corresponds to context update and requires that a specific contextual condition is true before continuing with the SAN traversal. The rightmost child node specifies the action to be taken in order to fulfil the goal. Rightmost node can also be a sub-goal node with its own three child nodes, or it can even be a construct joining several sub-goals in sequence or in parallel. As the SAN becomes more complex, involving several subgoals (Figure 1), it deepens and reveals its hierarchical and goal-directed characteristics. In this work, SANs are extended so that they can be used for detecting and deriving context from events.

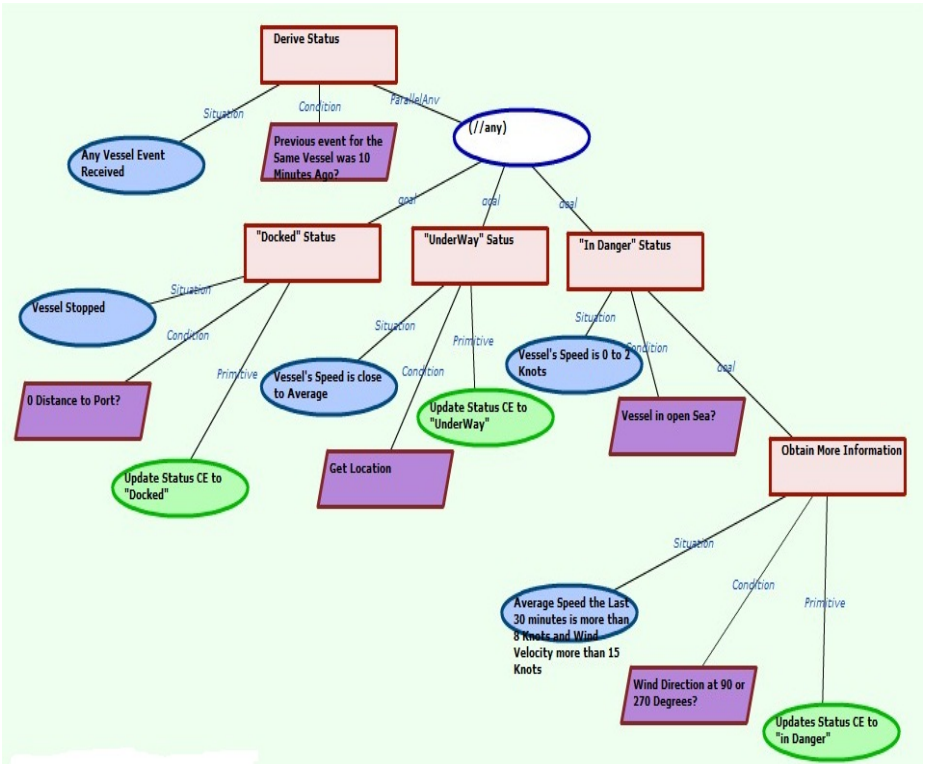

Fig. 1. SAN Illustration based on the marine vessel traffic scenario: Pink nodes denote Goals, Blue nodes denote Situations, Magenta nodes denote Context Condition and Green nodes denote Actions

This paper continues with a discussion about related work in the domain of event-based context management, while in section 3 it presents a generic context model that is considered appropriate for the needs of event marketplaces. In section 4, we consider two different approaches, one for detecting low level context and another

one for deriving higher-level contextual information using SANs. In section 5, we show how the generic context model can be specialized so that it can be instantiated to support an example scenario. We conclude in section 6 with a summary of our event-based context management approach.

2 Related Work

Context-awareness in service-oriented systems refers to the capability of a service or service-based application to be aware of its physical environment or situation and to respond proactively and intelligently based on such awareness; see e.g. [4]. Through the use of context, a new generation of service-based applications is expected to arise for the benefit of coping with the dynamic nature of the Internet; see e.g. [5], [6]. To reflect the varying nature of context and to ensure a universal applicability of context-aware systems, context is typically represented at different levels of abstraction [7]. At the first level of raw context sources there are context data coming from sensor devices, or user applications. At the next levels, context is represented using abstraction approaches of varying complexity. The work in [8] reviews models of context that range from key-value models, to mark-up schemes, graphical models, object-oriented models, logic-based models and ontology-based models. In [9] an ontological model of the W4H classification for context was proposed. The W4H ontology provides a set of general classes, properties, and relations exploiting the five semantic dimensions: identity (who), location (where), time (when), activity (what) and device profiles (how). The five dimensions of context have been also pointed out earlier in [10] where it was stated that context should include the 'five W': Who, What, Where, When, and Why.

Our work focuses on detecting context changes which correspond to either atomic or complex events and use complex event processing to model and identify them. Similarly to [11], we focus on events as a source of context because they are snippets of the past activities; therefore event processing may be viewed as a context detecting technology. Event processing results may be transferred to other applications, injecting context related information into services and processes. Based on the context definition of Dey and Abowd [1] and the associated five dimensions of context expressed in ontological model of the W4H [9], we define a high-level context model following an object-based modelling approach which can be easily specialized for different applications. We use semantic querying to extract contextual information from event payloads. Moreover, we exploit the reasoning capabilities of Situation-Action- Networks to enable dynamic derivation of context from multiple event streams and external services.

3 Context Model

We propose a context model as a stepping stone for facilitating event-based context detection and derivation functionality, in order to better understand situations in dynamic service oriented environments that demand for new additional information

sources or/and lead to a number of service adaptations as means for successfully coping with dynamic environmental changes. In order to achieve the goal of extracting contextual information, analyzing them and then deriving higher level context, we follow an event-based context modelling approach. In this section, we present such a Context Model (Figure 2), expressed in UML 2.0 class diagram. This model is based on the W4H model [9] that describes the five main elements associated within a context; the five elements are arranged into a quintuple (When, What, Where, Who, How).

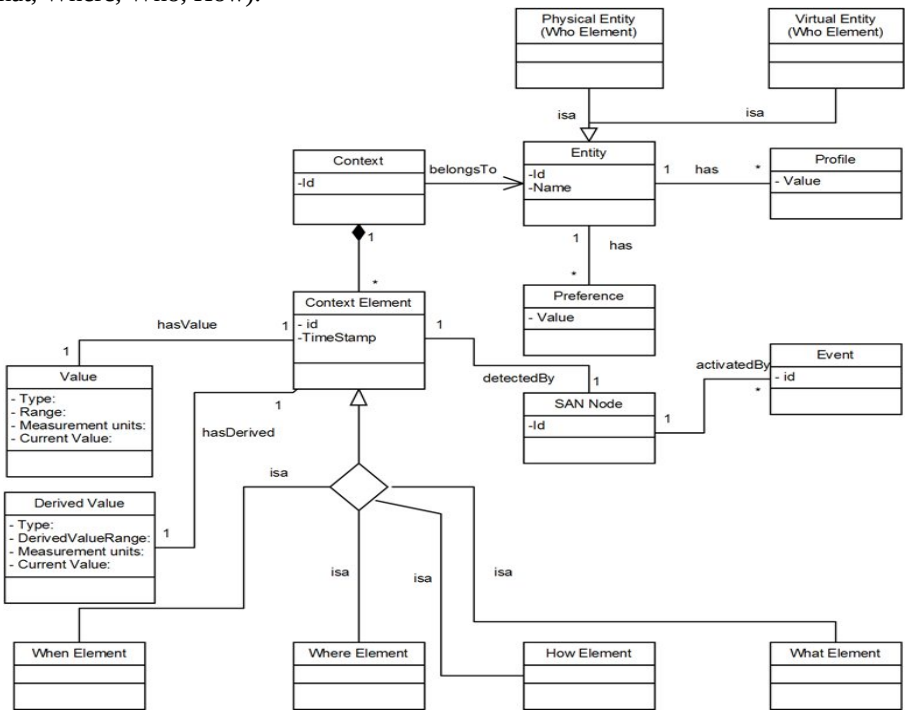

Fig. 2. Context Model

This Context Model expresses the temporal (i.e. When), spatial (i.e. Where), declarative (i.e. Who, What) and explanatory (i.e. How) dimensions of context having as central point of focus the notion of Entity. We refer to either physical or virtual entities with specific profiles and preferences that characterise them (e.g. vessel, port authority information system etc.). This way context obtains substance around the notion of an entity which can be a customer of an event marketplace system. The context class in our model constitutes the aggregation of several different context elements that may refer to five dimensions of context. Each Context element can have a value that can be acquired from the situation node of a SAN and/or a derived value that arises from any kind of reasoning process or call of external services. All context related information can be captured as objects which can store either a single scalar value or multiple values such as vectors, sets, lists etc. As any of the available context models [8], our model needs to become domain or application specific in order to be

useful. Next, we show how SAN Editor can be used to specialize and instantiate the generic context model.

4 Event-based Context Management

In our context modelling approach and implementation, we consider entities as being able to own SAN trees. The scope of context elements is distinguished into three levels: “Local”: Context elements can be updated and used only by a specific SAN instance. “Entity”: Context elements can be updated and used by any of the SANs owned by the same entity. “Global”: Context elements can be updated and used by all SANs independently to which entity they belong to.

Using the SAN Editor, we can perform context model specializations based on the application scenario and can formulate the necessary queries to events for extracting contextual information. We provide two approaches for acquiring context from simple or complex events and instantiating our context model. Both approaches use the SAN Editor for:

1. defining SPARQL queries to specific RDF event payload information that can update the values of an entity’s context elements; and
2. defining SANs that can use information from several event streams, analyse them and/or combine them with external services, in order to update the derived values of context elements. In this way, we succeed in acquiring higher level context compared to the lower level information that events carry.

The application of both approaches is presented in the following section through the marine vessel traffic illustrative scenario, which uses events related to marine traffic control that can be used to detect potentially dangerous vessel movements informing a controller when two vessels are approaching each other.

5 Illustrative Scenario

A vast amount of real time events are available from portals connected to automatic identification systems (AIS) that contain important vessel information worldwide (e.g., speed, course, vessel type, wind conditions etc.) and the several different users/authorities that might be interested in them. In order to exploit efficiently all these information in an automated way we use our context model and present how it can be specialized for the specific application domain while we give a glimpse to its possible run time instantiations.

Context Model Specialization: Our context model needs to become application specific in order to be useful. In this section, we focus on context model specialisation which pertains the definition of entities along with their context elements necessary for capturing the context in terms of a specific application scenario. We use the marine vessel traffic scenario which is related to vessel and marine traffic control observing systems.

In this scenario, we consider the entity Port Authority as the owner of all SANs discussed below while the entity of interest is the Vessel. In order to capture

contextual information related to Vessels' context, we have defined the following Context Elements that shape the specialization of our context model: Speed, Course, Position, Status, Distance2Port.

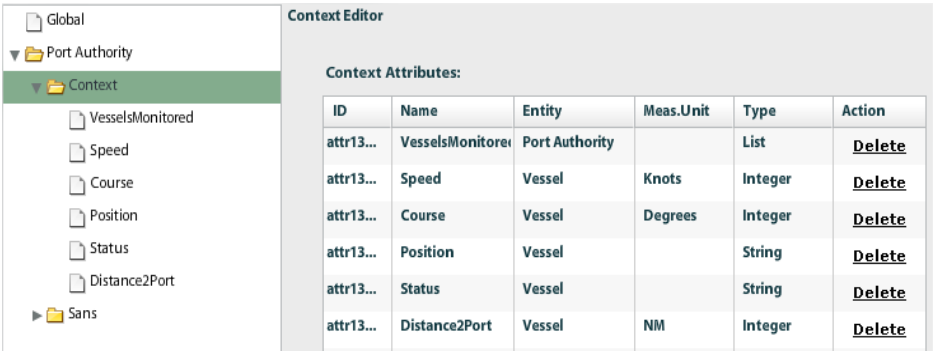

Fig. 3. Context Model Specialisation for the marine vessel traffic scenario

In Figure 3, the reader can find the complete list of the five context elements associated with the Vessel entity, specialising the context model for the marine vessel traffic scenario, using the SAN editor. This model specialisation will be instantiated at run time through the context detection and derivation approaches that are presented in the following sections.

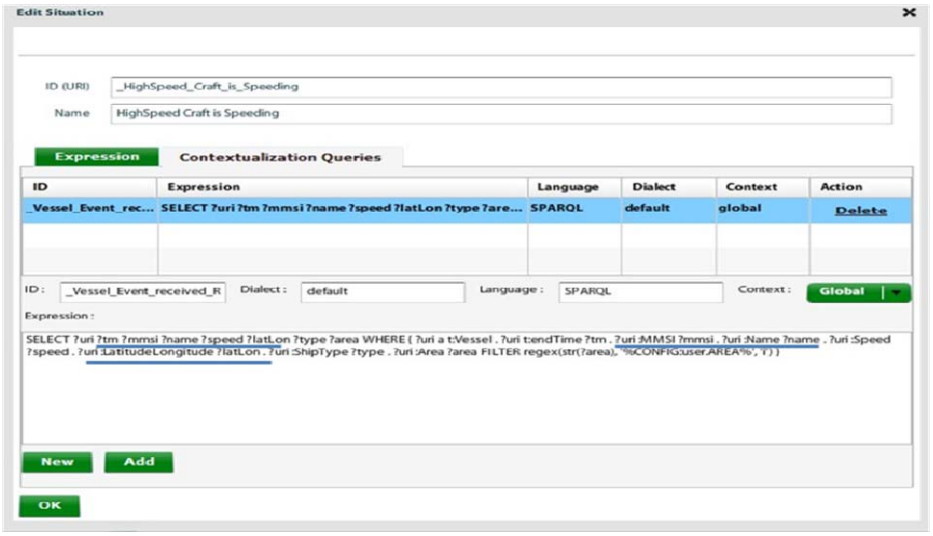

Fig. 4. SAN Editor Screenshot – Performing SPARQL Queries (about Position)

Detecting Context: In this section, we discuss our first approach for acquiring context from simple or complex events and instantiating our context model. Using SAN Editor, we are able to define SPARQL queries to specific event payload information that update the values of an entity's context elements. As we show in the

following figure 3 during our experiment we received events regarding a specific vessel called “Risoluto”. Details regarding the entity such as profile information automatically update the context of this entity based on the detected events in the situation node of a SAN.

Figure 4 depicts a screenshot of SAN editor with the required SPARQL queries for instantiating the “Position” context element of the vessel entity (Latitude/Longitude). Specifically, we query the vessel entity event payload with respect to the “LatLon” information. Similarly, other queries are used in the editor regarding the “Speed” and “Course” context elements and refer to event=based detection of low level context.

Deriving Context using SANs: Our second approach that we apply for extracting context from simple or complex events and instantiating our context model using SANs. We define a number of SANs that can use information from several event streams and combine them with external services in order to update the derived value class of context elements. In this way, we succeed in acquiring higher level context compared to the lower level information that events carry.

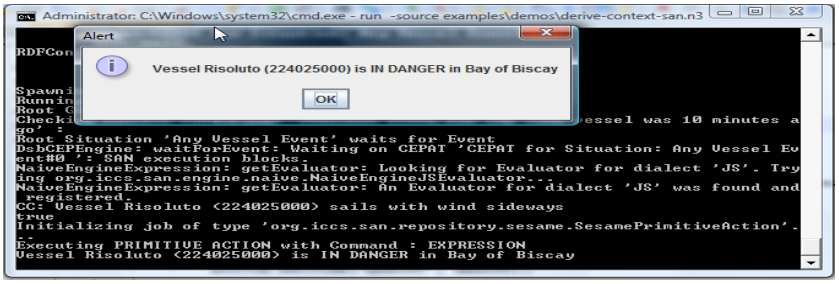

Fig. 5. SAN Engine Screenshot for Derived Status

This context derivation can be complex and may involve multi-level SANs. Figure 1 shows a SAN that upon traversal will be able to update the derived value class of the Status context element. Specifically, the status of the vessel becomes “Docked” whenever we detect a vessel that has been stopped and its distance from any port is close to zero or “UnderWay” whenever vessel’s speed is close to average and “In Danger” when the system realizes that the vessel has almost stopped (away from any port) and strong winds are blowing from the side. Figure 5 is a screenshot of the run-time execution of the specific SAN for deriving the vessel’s status. A pop up alert has been added in order to better demonstrate the context derivation regarding the Status context element.

6 Conclusions

In this paper we presented methods and tools for enhancing context detection and management based on events. This proposed context management approach presented here is considered appropriate for the needs of event marketplaces. We described a Context Model that was used by the developed mechanisms for performing event-based context detection and presented two different approaches for detecting low

level context (using SAN Editor) and deriving higher-level contextual information using Situation-Action-Networks (SANs). We provided with a meaningful context model specialization and demonstrated how simple or complex events coming from an event marketplace can be used and combined with external services, in order to derive higher level context with the use of SANs.

Acknowledgment

This work has been partially funded by the European Commission under project PLAY (Grant FP7-258659). The authors would like to thank the project partners for their advices and comments regarding this work.

References

1. Dey, AK & Abowd, GD: Towards a Better Understanding of Context and Context-Awareness, In Proceedings of the PrCHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness, pp. 304-307 (2000)
2. Stühmer, S. and Stojanovic, N.: Large-scale, situation-driven and quality-aware event marketplace: the concept, challenges and opportunities. In Proceedings of the 5th ACM international conference on Distributed event-based system, NY, USA, 403-40 (2011)
3. Verginadis, Y., Patiniotakis, I., Papageorgiou, N., Stuehmer, R.: Service Adaptation Recommender in the Event Marketplace: Conceptual View. R. Garcia-Castro et al. (Eds.): ESWC 2011 Workshops, LNCS 7117, pp. 194--201. Springer, Heidelberg (2011).
4. Abowd, GD, Ebling, M, Hunt, G, Lei, H & Gellersen, HW: Context-Aware Computing, IEEE Pervasive Computing, Vol 1, Issue 3, pp. 22–23 (2002)
5. Sheng, QZ, Nambiar, U, Sheth, AP, Srivastava, B, Maamar, Z & Elnaffar S: WS3: international workshop on context-enabled source and service selection, integration and adaptation, In Proceedings of the International Workshop on Context enabled Source and Service Selection, Integration and Adaptation, Beijing, China, pp 1263-1264 (2008)
6. Sheng, QZ, Yu, J & Dustdar, S: Enabling Context-Aware Web Services: Methods, Architectures, and Technologies, 1st ed., Chapman and Hall/CRC (2010)
7. Luther, M., Fukazawa, Y., Wagner, M., Kurakake, S.: Situational reasoning for task-oriented mobile service recommendation. The Knowledge Engineering Review 23(01), 7–19 (2008)
8. Bettini, C, Brdiczka, O, Henricksen, K, Indulska, J, Nicklas, D, Ranganathan, A & Riboni, D: A survey of context modelling and reasoning techniques, Pervasive and Mobile Computing, Vol. 6, Issue 2, pp.161–180 (2010)
9. Truong, HL, Manzoor, A & Dustdar, S: On modeling, collecting and utilizing context information for disaster responses in pervasive environments, In Proceedings of the 1st int. workshop on Context-aware software technology and applications, pp. 25–28 (2009)
10. Abowd, GD & Mynatt E.D.: Charting past, present, and future research in ubiquitous computing, ACM Transactions on Computer-Human Interaction, Vol. 7, Issue 1, pp. 29–58 (2000)
11. Etzion, O, Skarbovsky, I, Magid, Y, Zolotorevsky, N, & Rabinovich, E: Context Aware Computing and its utilization in event-based systems, Tutorial presented in DEBS, Cambridge, UK (2010)