Staying Aware in an Evolving World

— Specifying and Tracking Evolving Situations

Andrea Salfinger*, Werner Retschitzegger†, Wieland Schwinger‡
Department of Cooperative Information Systems
Johannes Kepler University Linz
Altenbergerstr. 69
4040 Linz, Austria
{andrea.salfinger*, werner.retschitzegger†, wieland.schwinger‡}@cis.jku.at

***Abstract*—In control center applications, such as road and air traffic management, a human operator's situation awareness (SAW) is of prime relevance. Operators need to assess critical situations emerging in that environment in order to undertake the appropriate (counter)actions, which becomes increasingly difficult w.r.t. today's rapidly changing and information-rich environments. Whereas automated SAW systems assisting operators are maturing, their focus is mainly on analyzing snapshots of the *current state* of the environment. Support for specifying and tracking *evolving situations*, however, is only partly dealt with, although being highly needed, as recent studies revealed. To address this issue, we therefore propose a conceptual *situation evolution model* (SEM) for rule-based SAW systems. This allows on the one hand to specify potential situation evolutions at design time, thereby exploiting domain experts' knowledge, and on the other hand serves as enabling mechanism to track actually assessed evolutions at runtime by model-based reasoning.**

I. Introduction

Situation Awareness. In control center applications, such as road and air traffic management, a human operator's situation awareness (SAW) is of prime relevance. SAW corresponds to the operator's adequate interpretation of the observed environment, gained through *Situation Assessment* (SA), i.e., the process of assessing relevant, potentially critical situations. Maintaining one's SAW of an evolving environment, however, also necessitates to *track* the *evolution* of these situations. To react accordingly, operators need to discriminate *emerging* situations from already *on-going* ones [1]. Furthermore, their assessment of a given situation depends on its past evolution, as well as its likely future development [2]. This highly complex task of *SAW maintenance* constitutes a difficulty for humans, as has been concluded in a user study on control center operators [1].

SAW Systems. Existing automated situation assessment systems (i.e., SAW systems), however, mainly focus on analyzing the *current state* of the environment by assessing on-going situations. Only few research efforts up to now have aimed at providing support for *maintaining SAW* by tracking these situations' evolutions, as we have shown in our previous survey on this topic [3].

Challenges of Situation Evolution. Tracking a situation's evolution (e.g., the development of the situation $s1$ "a massive traffic jam builds up after an accident") corresponds to monitoring and summarizing the evolution of its encompassed *objects* (e.g., the accident and the traffic jam): As objects involved in a situation evolve over time, their interrelations may change. Furthermore, additional objects may join a situation (e.g., a rear-end collision occurrs at the end of the traffic jam, thereby exacerbating the initially encountered situation $s1$), whereas other objects may disappear (e.g., the initial accident has been cleared, whereas the caused traffic jam still exists). Thus, the appearance of a specific type of object (e.g., the occurrence of a rear-end collision) may semantically correspond to a *new* situation in one case (e.g., a rear-end collision occurrs at a differently located traffic jam, corresponding to another situation $s2$), whereas in other cases, an already monitored situation should be just updated (e.g., the rear-end collision at the end of the built up traffic jam exacerbates the initially encountered situation $s1$, thereby requiring a different interpretation and reaction of the operator than in $s2$).

Contributions. In order to cope with such complex evolution scenarios, we propose a first step towards a dedicated *situation evolution model* (SEM), based on our experience with SAW systems for road traffic management (RTM) [4], which allows to specify and track situation evolutions of interest. This SEM serves on the one hand as a meta-model that conceptualizes how situations evolve over time, extending our previous work on situation evolution [5] and projection [6]. On the other hand, this SEM can be employed within established, rule-based SAW systems to allow for the *specification* and *tracking* of *evolving situations*. Overall, our approach supports SAW maintenance by enabling the operator to track different, evolving situations across their different evolution phases, ranging from their emergence to their clearance, and to inspect the interlinked current, past and potential future states of assessed situations w.r.t. their entire evolution path.

Structure of the Paper. In the next section, we propose a situation evolution model (SEM) for conceptualizing situation evolution. In Sec. III, we elaborate on how the SEM supports the tracking of situations at runtime. In Sec. IV, we describe a prototypical implementation of our approach, and compare our approach to related approaches in Sec. V. Finally, Sec. VI ends with an outlook on further potential applications of our approach w.r.t. *Situation Management*, as introduced by Jakobson [7].

II. Specifying Evolving Situations

In the present section, we introduce our *Situation Evolution Model* (SEM), a conceptual model for specifying and tracking evolving situations. To motivate the need for the specification of situation evolutions of interest and a corresponding tracking algorithm, we start by introducing a very simple example from the well-understood domain of road traffic management (RTM). As common in such event-driven systems, let us suppose that the occurrence of a wrong-way driver is encountered and reported to the system as shown in the lower part of Fig. 1, which is updated as additional events are received.

Situation. The appearance of a wrong-way driver represents a highly critical situation, which requires an operator's immediate, full attention and proper reaction. However, the operator must also be instantly alerted if a wrong-way driver approaches a tunnel, as this situation demands further, situation-specific counteractions: As a primary response, the tunnel should be closed for incoming traffic by setting the traffic lights, variable message signs should display according warnings. Furthermore, accidents inside tunnels necessitate different types of rescue operations compared to accidents occurring at unobstructed road segments, which needs to be taken into account when informing the emergency units. Therefore, due to the high criticality and time pressure, the operator requires according support by the SAW system, allowing to track and project this situation's development. Since the wrong-way driver's distance to the tunnel is proportional to the time horizon for undertaking the required actions, this scenario illustrates an evolving situation. Furthermore, the situation may evolve differently depending on whether there exists an exit in front of the tunnel, such that the wrong-way driver could leave the motorway prior to reaching the tunnel.

A. Our SAW framework at a glance — BeAware!

To support the assessment of such a priori specifiable situations, rule-based SAW systems basing on the JDL situation model [8], [9] have been considered as a suitable choice [1]. According to the JDL situation model, situations describe a specific set of interrelated objects (e.g., a wrong-way driver that is commensurate to a tunnel). In order to assess situations on objects observed at runtime, such SAW systems require a proper configuration by specifying *situation types* (STs), i.e., the operator needs to define which *relation types* between which *object types* are of interest. In our previous work on the generic, rule-based SAW framework BeAware! [4], we have elaborated on the application of various spatio-temporal primitive relation types (e.g., Allen's temporal relations [10], the Region Connection Calculus for spatial relations [11]), in order to model arbitrarily complex relation types of interest by combining such formally defined relation calculi. Thus, a part of the evolving example situation could be specified in the following fashion, which describes the ST where the wrong-way driver is in a *commensurate* distance to the tunnel:

$$WDCommTunnel := (\{w : WrongwayDriver, t : Tunnel\}, \{\mathcal{R}^{Distance}_{Commensurate}(w,t)\}), \quad (1)$$

whereby $R^{RelationFamily}_{Relation}(x,y)$ denotes a relation type of a specific relation family. These STs are translated to rules, and during SA, the descriptions of objects observed from the environment are matched against these rules. Objects that match a specific rule thus trigger the creation of a situation instance of the corresponding type.

However, this JDL-based, state-oriented approach to situation modeling [7] does not provide means to encode changes of the situation over time, which has already been criticized [1]. As a situation's criticality varies across its lifetime, we therefore took a first step towards conceptualizing the different evolutionary phases of a situation in our previous work [5]. Therein, we proposed to structure situation evolution into a *trigger* phase, corresponding to the emergence of a situation that may potentially develop to a critical one, a *climax* phase corresponding to a highly critical situation, and finally its *clearance* phase. However, our approach required the specification of a dedicated ST for each of these phases, thereby, yielding a different situation instance for each matched ST. Consequently, this approach did not enable operators to track a single situation instance from its emergence to its clearance. This hinders *investigative* Situation Management [7], which bases on a retrospective analysis of past situations, and situation learning in general, as it does not support analysis and conclusions on encountered situation evolutions (e.g., such as determining the overall lifespan of a given situation). Therefore, we identify a need to provide a means for specifying and tracking evolving situations across their entire lifetime, spanning their *trigger* phases to their *clearance* phases. Furthermore, we have not elaborated on how to track a situation's evolution across varying member objects, as further objects may join a situation, whereas others may disappear.

B. Expanding BeAware!'s Situation Types

In order to enable operators to specify not only STs, but entire *situation evolution types* (SET), by denoting potential situation evolutions of interest, we need to expand our previous approach. Therefore, we propose to extend the approved state-based notion of situations in a fashion corresponding to Lambert's STDF model [12], i.e., by introducing *transitions* between situation states defined according to the JDL notion. The rationale behind our approach is that we modify Lambert's model by formalizing the potential situation transitions of interest, which are specified within the SET.

Thus, instead of specifying different STs describing the evolutionary phases, which yield different situation instances at runtime, the operator specifies different *Situation State Types* ($\mathcal{SST}$) within a single SET. A $\mathcal{SST}$ implements the JDL situation model and is formalized as

$$\mathcal{SST} := (\Omega, \rho), \quad (2)$$

whereby,

– Ω is a finite, non-empty set of *object references* ($\mathcal{OR}$). An object reference corresponds to an *object type* ($\mathcal{OT}$) referenced by an *alias*, such as *Wrong-way Driver* w or *Tunnel* t. The $\mathcal{OT}$

corresponds to an ontological description of the observed real-world object, which may have various properties. The purpose of the alias will be elaborated on in Sec. III.
– ρ is a set of n-ary relation types ($\mathcal{RT}^{RelationFamily}_{Relation}$) holding between these $\mathcal{OR}$s, i.e., $\rho : \Omega^n \rightarrow \{true, false\}$. Since we do not impose any restrictions on the relation types' arity, relation types may be

- unary, i.e., $\rho : \Omega \rightarrow \{true, false\}$, corresponding to filters on specific object types' attributes (denoted by $\mathcal{OT}.attribute$), such as $\mathcal{RT}^{Filter}_{Location} := (\mathcal{OT}.location == x)$,
- binary, i.e., $\rho : \Omega \times \Omega \rightarrow \{true, false\}$, for example $\mathcal{RT}^{Distance}_{Close}(\mathcal{OT}_1, \mathcal{OT}_2)$),
- or n-ary, describing multiple interrelated $\mathcal{OT}$s.

The employed $\mathcal{RT}$s can be taken from various *relation families*. These can base on well-defined relation calculi (as described above, such as [10], [11]), whose interrelations have been studied and summarized by *Conceptual Neighborhood Graphs* (CNGs), as described for instance in [13], [14], or may be arbitrarily defined domain-specific relations.

An operator should be enabled of quickly assessing a given situation's evolutionary phase, i.e., whether it corresponds to a trigger situation that is starting to develop towards a critical one, or whether it is a formerly critical situation in its clearance phase. Therefore, a $\mathcal{SST}$ has dedicated properties, such as its *evolutionary phase* (i.e., whether it corresponds to a *trigger*, *climax* or *clearance* $\mathcal{SST}$) and *criticality*[1], which for instance allows to filter or rank assessed situations accordingly.

C. Extending BeAware! to Situation Evolution Types

To allow for a specification of a situation's *evolution* along its entire lifetime, ranging from its preconditions to its clearance, we extend these single state-based notions by introducing *evolution transitions*, denoted by the formal symbol '$\rightarrow$', between the $\mathcal{SST}$s. These describe how a situation evolves from one $\mathcal{SST}$ to a succeeding $\mathcal{SST}$, i.e., concatenate the distinct $\mathcal{SST}$s to comprise an overall SET. A specific situation evolves from $\mathcal{SST}$ st_i to a subsequent $\mathcal{SST}$ st_{i+1}, if either one or multiple of its relations ρ change, or the object type composition Ω changes:

$$st_i \rightarrow st_{i+1} := (st_i.\rho \neq st_{i+1}.\rho) \vee (st_i.\Omega \neq st_{i+1}.\Omega \wedge st_i.\Omega \cap st_{i+1}.\Omega \neq \emptyset), \quad (3)$$

whereby $st_i.\Omega$ denotes the set of object references defined in the $\mathcal{SST}$ st_i and $st_i.\rho$ the set of relation types holding in st_i. The purpose of the last condition, $st_i.\Omega \cap st_{i+1}.\Omega \neq \emptyset$, will become evident in Sec. III, as at least a part of the situation's comprising objects need to overlap within a single evolution step in order to allow for a tracking of the situation. In this fashion, we summarize a situation's overall evolution by means of *qualitative* changes between its encompassed objects, thereby abstracting from fine-grained, quantitative changes. Thus, the overall evolution, i.e., the entire *Situation Evolution Model* (SEM) of a SET, instantiates a labeled State Transition System $(S, \Lambda, \rightarrow)$: Its states S correspond to the $\mathcal{SST}$s, whereas its transitions '$\rightarrow$' are triggered by the evolution of the $\mathcal{SST}$s' encompassed objects (Ω) and relations (ρ). Its labels denote the change sets between the encompassed object type sets Ω and relation type sets ρ, summarizing the qualitative change between two successive $\mathcal{SST}$s.

$$SEM := (S, \Lambda, \rightarrow), \quad (4)$$

whereby,

- S comprises the set of $\mathcal{SST}$s,
- $\rightarrow$ encodes the transitions, i.e., binary relations over S, thus $\rightarrow \subseteq S \times S$,
- $\Lambda : (\Omega, \rho)$ consists of change sets in Ω and ρ between two succeeding states, i.e., $(st_i.\Omega \setminus st_{i+1}.\Omega \rightarrow st_{i+1}.\Omega \setminus st_i.\Omega, st_i.\rho \setminus st_{i+1}.\rho \rightarrow st_{i+1}.\rho \setminus st_i.\rho)$, thereby highlighting those objects and relations that are only part of one state but not the other, outlining what has changed within the evolution step.

A potential SEM for possible evolutions of the situation *wrong-way driver approaching tunnel* described above is shown by means of a state transition diagram in Fig. 1. This general definition of *situation evolution*, which encompasses both, changing *object* sets and changing *relation* sets, allows to address the variability of real-world situations: Only the distance relation changes between the $\mathcal{SST}$s *WD Commensurate to Tunnel* and *WD Close To Tunnel*, as would be expressed by the corresponding label ($\mathcal{RT}^{Distance}_{Commensurate}(WD, Tunnel) \rightarrow \mathcal{RT}^{Distance}_{Close}(WD, Tunnel)$), which discretizes the wrong-way driver's continuous movement to qualitative statements, whereas Ω, the set of $\mathcal{OR}$s, remains equal (*wrong-way driver w* and *tunnel t*). Furthermore, the SEM allows to model the nondeterministic behavior of the underlying real-world, since the $\mathcal{OT}$ wrong-way driver may cause an accident in any of these $\mathcal{SST}$s, which is modeled by a transition into the $\mathcal{SST}$ *WD Causes Accident*.

D. Supporting the Modeling of Situation Evolution Types

At a first sight, the general modeling approach of our SEM seems to induce also a drawback regarding complex real-world situation types, as the state space spanned by the SEM may appear vast and cumbersome to model, i.e., may lead to *complexity explosion*. However, this fact is mitigated by introducing further constraints on the possible transitions '$\rightarrow$', which facilitates an automated, semi-supervised state space generation. Tools supporting the modeling of SETs should exploit epistemic knowledge on relation interdependencies, i.e., CNGs, (i) to verify the correctness of the specified SETs (e.g., by detecting impossible relation combinations or transitions), and (ii) to generate and/or prune the state space. We have already realized such concepts in our previous work on the situation projection support provided by BeAware! [5], [6]. For instance, if two moving objects are *Close*, they may become either *Commensurate* or *Very Close* in the next evolution step, but not *Very Far*. Since the transitions can be thus formally defined by relation changes, the user may

[1] A measure indicating the situation's severity, which may for instance be denoted by a ranking between 1 (lowest) to 10 (highest criticality).

only need to specify *trigger* (e.g., the $\mathcal{SST}$ *Wrong-way driver commensurate to tunnel*) and *climax* (e.g., the $\mathcal{SST}$ *Wrong-way driver in tunnel*) situations of interest, and leave the generation of the full state-space to a dedicated tool, which reduces necessary modeling efforts.

III. Tracking Evolving Situations

The most elaborate situation evolution model would be of limited use, if the specified evolution could not be tracked at runtime. Our SEM enables *evolution tracking* by employing rule-based SA, as we will elaborate on in this section. Whereas the SEMs of different SETs encode the operator's interest in and knowledge about *what could potentially happen*, the tracking algorithm needs to keep track of an *actual situation's evolution*, which thus corresponds to a *path* through its SEM. Thus, it determines the actual development across a set of potential evolutions.

A. Goal of Evolution Tracking

Tracking Object Evolution. The tracking of the objects' evolution represents the grounding for tracking a situation's evolution, which SAW systems typically realize by the lower-level information fusion components [15] prior to SA, which are therefore assumed as given and not the focus of our discussion. Following [16], the overall evolution of an *object* O can be denoted by a sequence of snapshots of its current states o, thus a sequence of object states described by

$$O :=< o_{t1}, \ldots, o_{tn} >, \tag{5}$$

where n denotes the number of observed snapshots (taken at time instants t_1 to t_n).

Tracking Situation Evolution. Analogously to the object evolution model stated in Eq. 5, a situation's evolution over its entire lifetime would be formalized as

$$S :=< s_{t1}, \ldots, s_{tn} >, \tag{6}$$

whereby each s_{ti} corresponds to an actually assessed *observed situation state* of a specific *observed situation* S. Updates of a situation are triggered by updates on its contained objects. An *observed situation state* s_{ti} is created if a set of object states o_{ti} of different objects O matches a $\mathcal{SST}$. This s_{ti} has a property $s_{ti}.\mathcal{SST}$ referencing the $\mathcal{SST}$ it has matched (which in turn is associated with the SET it is contained in), furthermore each o_{ti} is associated with the $\mathcal{SST}$'s $\mathcal{OR}$ it has matched ($o_{ti}.\mathcal{OR}$). The goal of *tracking* a situation's actual evolution at runtime is to determine the "path" through its SET, as pictured in Fig. 1, and, ideally, project its probable evolution (i.e., the most probable successor states of the presently observed state).

Tracking Examples. We will exemplify this evolution tracking w.r.t. the *wrong-way driver approaching tunnel* SEM, which is illustrated in Fig. 1. The SAW system receives a message on the appearence of a wrong-way driver $W1$. Its first status $w1_{t1}$ creates $s1_{t1}$, corresponding to the $\mathcal{SST}$ *wrong-way driver commensurate to tunnel* of the SEM *wrong-way driver approaching tunnel*. Another status update $w1_{t2}$ creates $s1_{t2}$, as it again matches the $\mathcal{SST}$ *wrong-way driver commensurate to tunnel*. The next status update $w1_{t3}$, however, matches the $\mathcal{SST}$ *wrong-way driver very close to tunnel* in $s1_{t3}$, i.e., the $\mathcal{SST}$ *wrong-way driver close to tunnel* has not been observed. $w1_{t4}$ matches the *climax* $\mathcal{SST}$ *wrong-way driver inside tunnel*, $w1_{t5}$ indicates that the wrong-way driver has passed the tunnel, before the next status update $w1_{t6}$ is an all-clear signal, leading to the *clearance* $\mathcal{SST}$ $s1_{t6}$ that denotes the situation's end. All these observed situation states need to be associated to a single overall situation $S1$. Other situations, however, may take a different path through the SEM. For instance another situation $S2$ may be started in the more critical $\mathcal{SST}$ *wrong-way driver close to tunnel*, before the wrong-way driver $W2$ recognizes his mistaking and leaves at the next exit, thereby update $w2_{t2}$ is the all-clear signal, leading to an actually observed situation evolution that is ended before criticality escalation.

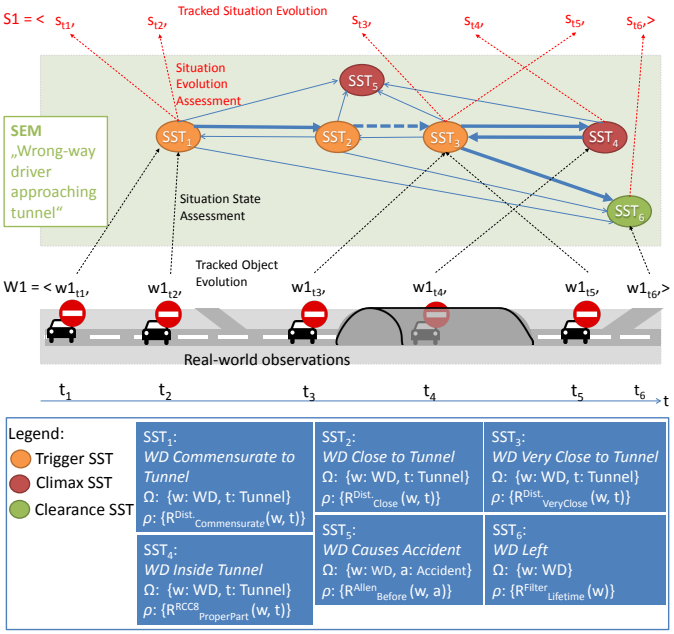

Figure 1. The tracking of an observed evolving situation w.r.t. its SEM.

Challenges of Evolution Tracking. Our algorithm for tracking a situation's evolution through its corresponding SEM must take into account the following issues:

- Situations may start in any $\mathcal{SST}$, not necessarily a *trigger* $\mathcal{SST}$, and end in any $\mathcal{SST}$, not necessarily a *clearance* $\mathcal{SST}$ indicating a resolved situation (e.g., the situation evolution may end when the wrong-way driver causes an accident). This may be due to incomplete information sensed from the environment, but also due to incompletely specified evolutions, as not every potential evolution will be known in advance, but learned at runtime.
- The set of involved objects Ω may change across the situation's evolution, i.e., $s_i.\Omega \neq s_{i+1}.\Omega$ for two succeeding situation states s_i and s_{i+1}.

- The SAW system may not be able to observe every transition of the evolving situation since it may receive incomplete information from the environment, for instance due to low sampling frequencies. This challenge is exaggerated by the fact that the number of skipped $\mathcal{SST}$s is typically unknown.

B. The SEM Tracking Algorithm

The SAW system's situation assessor component is triggered in regular intervals. Its input consists of the current object states o_i corresponding to the latest states of observed objects O, i.e., the current environmental snapshot sensed from the environment at time instant t_i. The goal is to assess situations S corresponding to situation instances of the defined SETs, whereby the assessor should create new situations, or update existing situations S by appending a new situation state s_i, accordingly. Since the inference of the potential evolution path should not impact the performance of SA by any means, we thus propose to split SA in two subsequent stages. By decoupling the assessment of the current situation states from determining the evolution paths in-between, the actual SA is thus not confounded by evaluating the potential evolutions.

(1) Situation state assessment. Situation states from the environment are assessed by matching the current object states o_i of the observed objects O against all $\mathcal{SST}$s of the defined SETs. If a $\mathcal{SST}$ is matched, an *observed situation state* is instantiated. Thus, this stage entirely conforms to SA as performed in most currently available, rule-based SAW systems. The output of this stage comprises assessed situation states s_i.

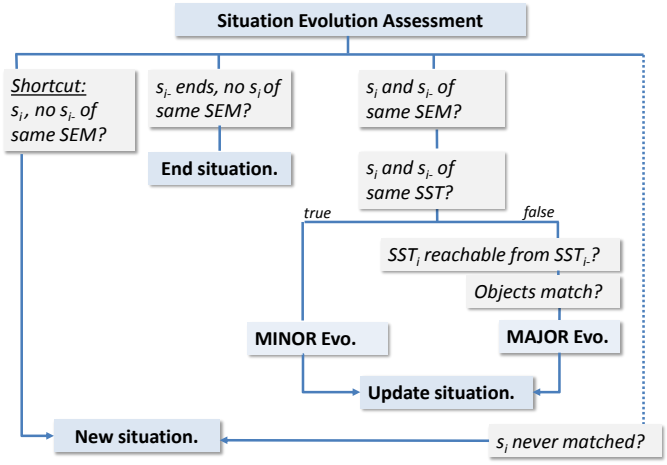

Figure 2. A flow chart summarizing situation evolution assessment.

(2) Situation evolution assessment. The second assessment phase is responsible for determining whether the currently assessed situation states s_i correspond to a *new* situation S, or denote an *update* to an already existing situation S, thereby assessing the situations' evolutions. The assessed situation states s_i are thus fed into the second assessment engine, which contains the situation states assessed in the previous runs, s_{i-}. Remember that each s is associated with the SEM that created it, as explained in the course of Eq. 6. The assessment of a situation's evolution boils down to the following steps:

(a) s_i exists, no old situation state s_{i-} of the same SEM available: In this case, a new situation S of the corresponding SET is instantiated, and the situation state s_i is added as initial situation state.

(b) s_{i-} exists, no new situation state s_i of the same SEM is found: In this case, the situation is finished, thus its lifetime is set to end.

(c) s_i and s_{i-} found, which are of the *same* $\mathcal{SST}$: In this case, it needs to be assessed whether all object states o_i of the new s_i refer to the same objects as the object states o_{i-} of the old situation, i.e., denote an identity or evolution of the same overall objects O, thus requiring *unification*, similar to the Rete algorithm [17], according to

$$\begin{aligned} &\forall o_i \in s_i.\Omega : \exists o_{i-} \in s_{i-}.\Omega : O(o_i) == O(o_{i-}) \\ \wedge &\forall o_{i-} \in s_{i-}.\Omega : \exists o_i \in s_i.\Omega : O(o_i) == O(o_{i-}), \end{aligned} \tag{7}$$

whereby $O(o_i)$ denotes the overall, evolving object O associated with object state o_i according to Eq. 5.

Thus, if both situation states refer to the same objects, an existing situation S needs to be updated. This case is termed *minor evolution*, which denotes the case where the objects of a situation have evolved, however in a way that their interrelations did not change. Since we want to be able to persist a situation's entire evolution, such small changes have to be considered as well (e.g., the SAW system has received an update on the wrong-way driver's current position, however, its distance relation to the tunnel is still *commensurate*).

(d) s_i and s_{i-} found, which are of the same SEM, but of different $\mathcal{SST}$s: This case corresponds to a *major evolution*, denoting a situation whose encompassed objects have evolved in a way that their relations have changed (e.g., another update on the wrong-way driver's current position has changed its distance relation to the tunnel from *commensurate* to *close*), indicated by a transition between distinct $\mathcal{SST}$s. In order to assess whether s_i, matched from $\mathcal{SST}_i$, represents an evolution from a previously recognized $\mathcal{SST}_{i-}$, i.e., s_{i-}, the situation assessor needs to possess the knowledge about the valid evolution paths. Thus, it must essentially *know* the corresponding SEM, which describes the valid transitions. In order to determine whether an evolution of $\mathcal{SST}_{i-}$ to $\mathcal{SST}_i$ is valid according to the specified model, we solve this evolution tracking by performing *model-based reasoning* [18] on the SEMs. Thus, prior to performing *object unification*, the situation assessor first performs a *reachability analysis* on the corresponding SEM, to assess whether transitioning from $\mathcal{SST}_{i-}$ into $\mathcal{SST}_i$ is possible. If this is the case, then *object unification* can be performed, in which it is evaluated that both situation states refer to the same objects that should have remained within the situation during its evolution. Thus, the *unification* must be performed on the object identities of the $\mathcal{SST}$s' *overlapping object references* Θ, defined as

$$\Theta := (\mathcal{SST}_{i-}.\Omega) \cap (\mathcal{SST}_i.\Omega). \tag{8}$$

Unification is performed as:

$$\begin{aligned} \forall(o_i \in s_i.\Omega \wedge o_i.\mathcal{OR} \in \Theta) : \\ \exists o_{i-} \in s_{i-}.\Omega \wedge o_{i-}.\mathcal{OR} \in \Theta : \\ o_i.\mathcal{OR} == o_{i-}.\mathcal{OR} \wedge O(o_i) == O(o_{i-}). \end{aligned} \tag{9}$$

In order to be able to uniquely refer to specific objects across $\mathcal{SST}$s, we require that the distinct $\mathcal{OT}$s of Ω need to be distinguished by *aliases* in Eq. 2, to enable the operator to specify which objects have evolved. In the *unification* step, objects whose $\mathcal{OR}$s are referred to in Θ are bound to these aliases, and checked for identity between equal aliases of the objects in $s_i.\Omega$ and $s_{i-}.\Omega$.

If no existing s_{i-} can be found in the update steps (3) and (4) that matches s_i, a new situation S is instantiated for s_i.

IV. Prototypical Implementation

In the present section, we provide a short overview on our prototypical implementation of our approach. We based our implementation on Java, a PostgreSQL (www.postgresql.org) database as knowledge base, including the PostGIS (postgis.refractions.net) extension to allow for high-performance spatial computations, and the JBoss Drools (www.jboss.org/drools) rule engine.

We employ two different rule engines, more specifically stateful knowledge bases, which thus maintain their state during the invocations. Therefore, the first knowledge base, which performs the situation state assessment, can be initialized with static (e.g., environmental infrastructure) data. The SA process is periodically triggered in user-specifiable intervals, whereby the rule engines are subsequently fired according to the procedure described in Sec. III. In each assessment run, the first knowledge base is populated with facts corresponding to the state of objects observed from the monitored environment. An object state remains valid until an updated state of the respective object, denoted by a unique identifier, is inserted, in which case its corresponding fact handle is updated, or the denoted lifetime of the object ends, in which case it is retracted from the knowledge base. Each rule of the first rule engine corresponds to the description of a specific $\mathcal{SST}$, in which the observed object states are matched towards the sought-after $\mathcal{OT}$s and their corresponding relations. Matched rules trigger the instantiation of a situation state of the corresponding $\mathcal{SST}$. After rule execution, all these situation states are inserted as facts into the second knowledge base, which is then fired. If an object of the first knowledge base is retracted, due to its expired lifetime, it is checked whether it has been defined as *situation clearance object* for any SET, and has been participating in any situation states of such SETs. If this is the case, we generate a dedicated situation state denoting this situation's end, which is also fed into the second knowledge base.

The second knowledge base consists of four general rules only implementing the algorithm proposed in III-B. In the matching procedure of the two rules for updating an existing situation instance, a dedicated component performs model-based reasoning on the situation states' SET objects in order to perform *reachability analysis* and *object unification*. If a specific situation is updated, then its previous situation state is retracted. Thus, one situation state per situation instance is kept in the knowledge base. A situation is set to end if a situation ending state is received, in which case the respective situation instance is retracted from the knowledge base. Therefore, the evolution specification is solely comprised in the SET objects, whereas it is not necessary to create any specific rules for tracking a certain SET.

V. Related Work

Although it has been recognized that tracking evolving situations represents a crucial task in SAW systems, (c.f., [7], [19], [20], [21]), only few concrete approaches addressing this issue have been suggested, as encountered in our recent survey on the evolution support of SAW systems [3]. In the present section, we will discuss these approaches, starting with the most closely related ones.

A first step towards specifying and tracking evolving situations over their entire lifetime in a rule-based SAW system has been undertaken in [22]. Operators can specify a temporally ordered sequence of relations between objects, depicting how these relations change over time. However, the system detects these situations *after* they have completed their evolution, instead of monitoring evolving *on-going* situations. Assessing an *intermediate* state within an *evolving* situation is not supported, which has been also identified as necessary future work by the authors themselves.

Pereira et al. [23] have elaborated on means for situation lifecycle management in rule-based SAW systems. Their implementation-focused approach supports activation, state-maintenance and deactivation of a situation instance. Whereas we have employed similar situation deactivation strategies in our prototypical implementation described in Sec. IV, which also bases on the JBoss Drools platform, our approach extends to detecting and managing a situation's lifecycle across different evolutionary phases, corresponding to different rules. In our approach, a situation instance is allowed to match different rules across its lifecycle, for which we propose an explicit tracking algorithm. Furthermore, since situations in our application domain consist of objects evolving themselves, our situation deactivation strategies allow to keep track of the object's encompassed in a situation. If specific objects disappear, which have been registered to *end* specific situations, our approach takes into account that all situations referring to these objects need to be finished either.

Lambert's State Transition Data Fusion (STDF) model [16] denotes snapshots of the environment as states, and considers changes over time as transitions between these states. Regarding situation evolution tracking, Lambert identifies the problem of how to distinguish an *initiation* process, corresponding to the detection of a novel situation, from an *update* process which delivers a new state for an existing situation, which he solves with a *Bayesian approach*. Thus, evolution tracking is solved in a probabilistic fashion, by evaluating the most likely evolution sequence, and does not allow for an explicit

specification of evolving situation types of interest, as is the focus of our approach.

Meyer-Delius et al. [24] model situations as Hidden Markov Models (HMM), which depict a situation as a sequence of states. Regarding situation evolution tracking at runtime, the likelihood of sequences of observations from the environment is evaluated for each trained situation model, whereby the model that provides the best explanation for the given sequence is chosen. Thereby, whereas the operator can specify the structure of the HMM, the successful tracking depends on a representative set of training data. Furthermore, since each situation instance is evaluated independently from the others, the complexity of the situation tracking algorithm increases linearly with the number of situation types and the number of objects (that are assessed w.r.t. the situation types).

VI. Future Work

In the present section, we outline our plans for future work, in which we aim at supporting the modeling of SETs and studying means of learning from actually tracked situations.

Support for Specifying SEMs. To support operators in the specification of SEMs, we are currently developing a *Situation Type Editing Suite*, which should facilitate the specification of SETs, as sketched in Sec. II-D. Our tool aims at supporting the operator-guided specification of SETs, which are verified for syntactical and semantical correctness, and generates the corresponding rules.

Learning from Tracked Situation Evolutions. We further advocate that *tracking* a situation's *evolution* may not only address the online information requirements of human operators, but could also be exploited to refine the SEMs. *Persisting* the tracked evolutions of observed situations could allow for learning from previously observed situations, which represents a still under-exploited aspect in currently proposed SAW systems [3], such as determining the SEM's transition probabilities. *Modeling* and *tracking* the overall evolution of an observed situation from its *preconditions* to the criticality escalation leading to the climax phase and its final clearance could allow to "reuse" knowledge from previously found situations to refine situation assessment, the situation models, and the situation predictions in the future: (i) Storing a situation's preconditions (i.e., the trigger situation) would help to detect emerging situations in the future, which allows to undertake preventive action before the situation reaches its climax and becomes too critical, thus allows for *Investigative Situation Management* (SM) [7]. (ii) Keeping the evolution path may help to predict a current situation's evolution based on previously observed situations, thus supports *Predictive SM*. (iii) If the undertaken actions are also tracked, operators may assess which actions led to the fastest or most desirable resolution of past situations, thereby supporting *Situation Resolution*.

VII. Acknowledgments

This work has been funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under grant FFG FIT-IT 829598, FFG BRIDGE 838526, FFG BRIDGE 832160, WTZ/ÖAD AR18-2013 and UA07-2013.

References

[1] M. Nilsson, J. van Laere, T. Ziemke, and J. Edlund, "Extracting rules from expert operators to support situation awareness in maritime surveillance," in *Procs. of the 11th Intl. Conf. on Information Fusion*, 2008.

[2] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, no. 1, pp. 32–64, 1995.

[3] A. Salfinger, W. Retschitzegger, and W. Schwinger, "Maintaining situation awareness over time - a survey on the evolution support of situation awareness systems," in *Conf. on Technologies and Applications of Artificial Intelligence (TAAI 2013)*, 2013.

[4] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, and W. Schwinger, "Beaware!—situation awareness, the ontology-driven way," *Intl. Journal of Data and Knowledge Engineering*, vol. 69, no. 11, pp. 1181–1193, 2010.

[5] N. Baumgartner, W. Retschitzegger, W. Schwinger, G. Kotsis, and C. Schwietering, "Of situations and their neighbors—evolution and similarity in ontology-based approaches to situation awareness," in *Proc. of the 6th Intl. and Interdisciplinary Conf. on Modeling and Using Context (CONTEXT)*. Springer, 2007, pp. 29–42.

[6] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, and W. Schwinger, "Situation prediction nets," in *Conceptual Modeling – ER 2010*, ser. LNCS. Springer Berlin Heidelberg, 2010, vol. 6412.

[7] G. Jakobson, J. Buford, and L. Lewis, "A framework of cognitive situation modeling and recognition," in *Military Communications Conf., 2006. MILCOM 2006. IEEE*, 2006, pp. 1–7.

[8] A. N. Steinberg, C. L. Bowman, and F. E. White, "Revisions to the JDL data fusion model," vol. 3719, pp. 430–441, 1999.

[9] J. Llinas, C. Bowman *et al.*, "Revisiting the JDL data fusion model II," in *Procs. of the 7th Intl. Conf. on Information Fusion (FUSION 2004)*, 2004, pp. 1218–1230.

[10] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, pp. 832–843, 1983.

[11] D. A. Randell, Z. Cui, and A. G. Cohn, "A spatial logic based on regions and connection," in *Computer*. Morgan Kaufmann, 1992, vol. 92.

[12] D. Lambert, "STDF model based maritime situation assessments," in *Procs. of the 10th Intl. Conf. on Information Fusion (FUSION 2007)*, 2007.

[13] C. Freksa, "Conceptual neighborhood and its role in temporal and spatial reasoning," in *Decision support systems and qualitative reasoning*. North-Holland, 1991, pp. 181–187.

[14] A. Gerevini, "Combining topological and size information for spatial reasoning," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 1–42, 2002.

[15] E. L. Waltz and J. Llinas, *Multisensor Data Fusion*. Norwood and MA and USA: Artech House, Inc, 1990.

[16] D. Lambert, "A unification of sensor and higher-level fusion," in *Procs. of the 9th Intl. Conf. on Information Fusion*, 2006.

[17] C. L. Forgy, "Rete: a fast algorithm for the many pattern/many object pattern match problem: Expert systems." IEEE CS Press, 1990.

[18] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall Press, 2009.

[19] L. Niklasson, M. Riveiro *et al.*, "Extending the scope of situation analysis," in *Procs. of the 11th Intl. Conf. on Information Fusion*, 2008.

[20] C. Matheus, M. Kokar *et al.*, "SAWA: An assistant for higher-level fusion and situation awareness," in *Proc. of SPIE Conf. on Multisensor, Multisource Information Fusion*, 2005, pp. 75–85.

[21] ——, "Lessons learned from developing SAWA: a situation awareness assistant," in *Proc. of the 8th Intl. Conf. on Information Fusion*, 2005.

[22] J. Edlund, M. Grönkvist, A. Lingvall, and E. Sviestins, "Rule-based situation assessment for sea surveillance," *Proc. SPIE 6242, Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, vol. 6242, 2006.

[23] I. Pereira, P. D. Costa, and J. Almeida, "A rule-based platform for situation management," in *Procs. of the IEEE Intl. Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA 2013)*, 2013, pp. 83–90.

[24] D. Meyer-Delius, C. Plagemann, and W. Burgard, "Probabilistic situation recognition for vehicular traffic scenarios," in *Procs. of the IEEE Intl. Conf. on Robotics and Automation (ICRA '09)*, 2009, pp. 459–464.