An Approach to Integrated Cognitive Fusion

G. Jakobson
Altusys Corp
52 Vernon Street
Brookline, MA 02446
USA
jakobson@altusys.com

L. Lewis
S. New Hampshire University
2500 North River Road
Manchester, NH 03106
USA
l.lewis@snhu.edu

J. Buford
Altusys Corp
52 Vernon Street
Brookline, MA 02446
USA
buford@altusys.com

Abstract – *We describe the integration of two technologies to achieve cognitive fusion--the dynamic analysis of data combined from multiple sources in order to recognize complex dynamic situation patterns, construct models or hypotheses of unfolding situations, and take action in response to situations. The two technologies are temporal event correlation and case-based reasoning. We describe both technologies, present an integrated architecture, and discuss implementation issues. The goal of the research is to develop a decision-support system for situation and decision awareness, where domains include battlefield management, homeland security, environmental sensing, crisis management, telecommunications management and other dynamic and information-rich domains.*

Keywords: Cognitive fusion, information fusion, situation awareness, event correlation, case-based reasoning, battlefield management.

1 Introduction

The focus of this paper is on cognitive fusion within the context of analysis and reasoning about dynamic situations. In particular, we are concerned with situations such as those encountered in the management of a battlespace, surveillance of complex technological systems, and mobilization of countermeasures in real-time emergency situations in health care and homeland security applications. These applications involve a large number of dynamic objects that change their state in time, and are engaged in complex spatio-temporal relations. From the management viewpoint it is important to understand the situations in which these objects participate, to recognize emerging trends and potential threats, and to undertake protective actions that lead to predefined goal situations.

Although information fusion has been in the focus of intensive research, particularly in the defense community, our approach emphasizes two important aspects of information fusion, cognitive information processing and fusion of information in dynamic real-time situations. We define *Cognitive Fusion* as a process of multi-source data fusion, where a qualitatively new meaning is assigned to the fused data. We associate three basic functional qualities with *Cognitive Fusion*:

(i) Situation Awareness
(ii) Decision Awareness
(iii) Knowledge Awareness

Situation Awareness is a result of multiple information processing functions, including:

- Understanding the meaning of multi-media data, recognizing complex spatial and time-dependent patterns, and constructing models of situations
- Determining threats, intrusions, hostile activities, system failures, and other activities that reveal intent or capability.

Decision Awareness is based on the following functions:

- Reasoning about situations, planning and implementing actions or action-oriented decisions as part of optimal management plans
- Informing and advising commanders regarding the potential ramifications of the suggested actions.

Knowledge Awareness deals with:

- Mining historic operational data, generating new fusion patterns, discovery of new situation classes
- Learning, and improving the skills and effectiveness of fusion procedures, situation analysis, and decision-making procedures.

As a cognitive function we are considering Cognitive Fusion as a discipline having its roots in Cognitive Information Processing [1], where the core human cognitive functions are modeled, including reflection, reasoning, learning, explanation, and communication. The term cognitive fusion was used in [2] to denote the fusion of multi-sensor imagery based on concepts derived from neural models of visual processing and pattern recognition. In [3] the principles of cognitive fusion were applied to enhance the computational algorithms of moving target recognition.

The above-mentioned cognitive fusion functions could be mapped to the Level 2 and Level 3 functions of the JDL Fusion Model [4, 5].

Real-time information fusion deals with time-dependent events and dynamic situations. Modeling dynamic situations has been the research focus of several scientific disciplines, including human factors and artificial intelligence. Informally, situations were considered as snapshots of the world at some time instant, while a strict formal theory offered by artificial intelligence research was based on non-monotonic reasoning [6]. The human factors community coined the

term "situation awareness" [7]. In recent years the notion of situation awareness has found a prominent place in defense related programs, mostly related to battlefield management, information fusion, and analysis of sensor data [8-10]. An ontology of typical battlefield situations was proposed along with their specification in UML [11].

Our approach to cognitive fusion in dynamic situations is based on integration of two artificial intelligence disciplines in which the authors have been involved for several years: real-time event correlation (EC) [12] and case-based reasoning (CBR) [13]. EC is a widely recognized approach for telecommunication network root cause fault analysis and CBR is an effective paradigm for reasoning and decision support in applications such as health care, diagnostics, and law.

This paper is organized as follows. Sec. 2 examines the role of cognitive fusion in the overall architecture of fusion, and describes the situations and the dynamic aspects of situation transition. Sec. 3 discusses the principles of modeling situations with CBR. Sec. 4 presents the model of information fusion based on real-time event correlation. Sec. 5 examines the architecture of cognitive fusion based on distributed component services, and Sec. 6 outlines future research directions.

2 Cognitive fusion in dynamic situations

2.1 Generalized fusion model

Fig. 1 shows a generalized two-dimensional fusion model. The horizontal dimension shows the three fusion functional areas– analysis, reasoning, and acting, and the

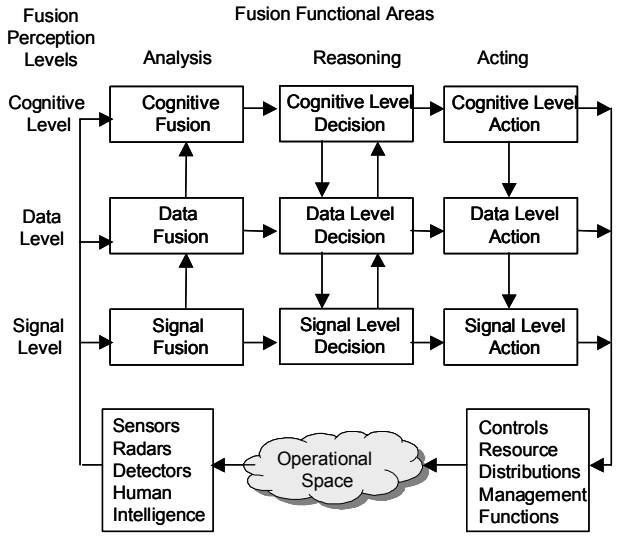

Fig. 1. Generalized two-dimensional fusion model

vertical dimensions shows the three fusion perception levels - signal, data, and cognitive levels. The functional areas reflect the fundamental processes of purposeful intelligent behavior: (i) to understand individual features and entities (objects), object associations, and complex dynamic situations; (ii) to reason about the object associations and situations in order to predict changes in situations and plan required actions, and (iii) to implement the planned actions, including communication with other systems and humans. Each of these processes is present in the three levels of fusion perception, although they are defined with different levels of abstraction, e.g., the analysis process at the signal level mostly deals with the feature extraction and recognition of simple objects; at the data level the analysis processes aims at recognition of complex individual objects and core object associations, while at the cognitive level the task is to analyze complex dynamic situations.

The three fusion processes form a closed dynamic operational space management loop (see Fig. 1). Depending on the objectives, the management of the operational space might proceed at a lower level of abstraction, e.g., at the data or signal level. In this paper our interest is on cognitive level fusion.

2.2 Situations

Informally, a situation is a state of a set of entities at some particular time. It could be a vector of values of attributes belonging to one entity or multiple entities. A situation could also be the state of the relation(s) between one or multiple entities at some particular time. A simple situation could be formed by one entity, where one attribute, for example location, is changing by forming a series of distinct situations at each consequent time moment. For example, a tank at a specific location performing a specific task at a specific time is a situation. In this example, one of the parameters of the situation will be Agent_1, whose class type is TANK. The class TANK is an abstract entity of the domain.

Complex situations are formed over the collection of entities engaged in multiple relations. Complex situations may also be constructed from component situations using Boolean operators. One important factor in the construction of situations is time management: entities comprising a situation as well component situations in a complex situation should comply with the unified time requirement, i.e., the time values for all attributes of all entities involved in a situation should fall into a predefined time interval.

The entities could be active ones, i.e., entities which change their attribute values over the time, or passive ones whose attribute values are considered constant over the observed time period. An active entity is also characterized by the feature of appearance (possibly – creation) or disappearance (possibly – destruction) at a certain time moment. In some battlespace management tasks it is important to model situations where appearance and disappearance facts are associated with static entities. Entities could participate in different class, structural, containment, connectivity, spatial, organizational and multiple other domain specific relations, either in binary or higher order relations. For example,

Entity1 SUBCLASS Entity2
Entity1 LOCATED_AT Entity2,
Entity1 ABOVE Entity2

Entity1 SUBORDINATE_TO Entity2
Entity1 TARGETS Entity1
Entity1, Entity2, Entity3 FORM_UNIT Entity4

Each relation has its own attributes, including time. In addition to the domain specific attributes, the relations might have algebraic properties, since often, for logical reasoning purposes, it is important to know whether these relations are reflexive, transitive, or symmetric.

Attributes in entities may have attached procedures, which enforce constraints associated with attribute values or implement computational or symbolic procedures over the attribute values of the native or remote entity.

2.3 Situation transitions

How situations change is an important aspect of cognitive fusion modeling. Such changes can be represented by a situation transition graph (STG) as shown in Fig. 2. Each node in an STG represents a situation. Directed arcs represent the transitions from one situation to another situation. For example in the battlespace context, the dynamics of the battlespace are represented by an STG with transitions from one battlespace situation to another. During this modeling process certain situations are identified as the start, target, undesirable, and transitional situations. Other types of situations could be introduced depending on the objectives of the modeling process. The importance of such transitions is that the transitions may be preconditioned by events, which are fused from multiple events sources. We call these fusion-driven situation transitions.

Fig. 2 also depicts case-base reasoning (CBR) driven situation transitions. These transitions, taking advantage of the analogical reasoning power of CBR, project potential future situations, which might occur in the

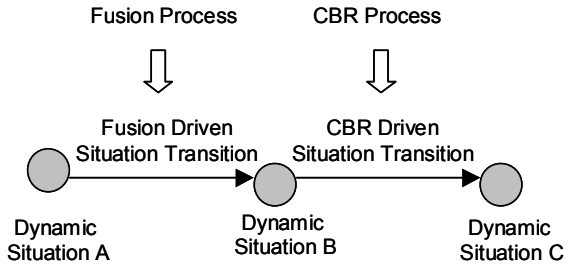

Fig. 2. A situation transition graph: transitions defined by fusion and CBR

managed operational space. Such projection of future situations is a key feature in implementing the functions of situation awareness and threat analysis. We will return to the mechanism of the CBR driven situation transition in Sec. 5.1. Both situation transition methods are utilized in the fusion analysis and reasoning functional areas as shown in Fig. 1.

3 Modeling situations with CBR

Fig. 3 shows the high-level model of CBR. In general, a set of events is posed to the CBR system, whereupon four processes are carried out. First, the set of events is compared to a library of former situation templates, and a set of maximally similar cases is retrieved. Thus, the inputs for the Retrieve module are a set of events and a case library. In the CBR literature, a number of retrieval algorithms have been proposed. These are listed as a set of design options attached to the Retrieve module in Fig. 3. The simplest and weakest algorithm is key-term matching, whereas the most complex but strongest algorithm is analogy-based matching.

The case library can be thought of as a set of former experiences with situations that are potentially similar to the situation at hand. However, it is hardly ever the case that a former situation will be exactly like the current situation. Typically a former situation has to be tweaked in some way to render it applicable to the nuances of the current situation. This is the task of the Adapt module in Fig. 3, where several design options of the module are listed. Adaptation by substitution covers those episodes in which an object that occurs as a descriptor in the current situation should be substituted throughout for an object that occurs as a descriptor in the retrieved case.

The Execute module is straightforward. The user may choose to act on the retrieved/adapted situation. The execution may be conducted manually or may be carried out automatically by the decision-maker, either in supervised or unsupervised mode. In addition, the execution of an action or plan may involve cooperation with other individuals.

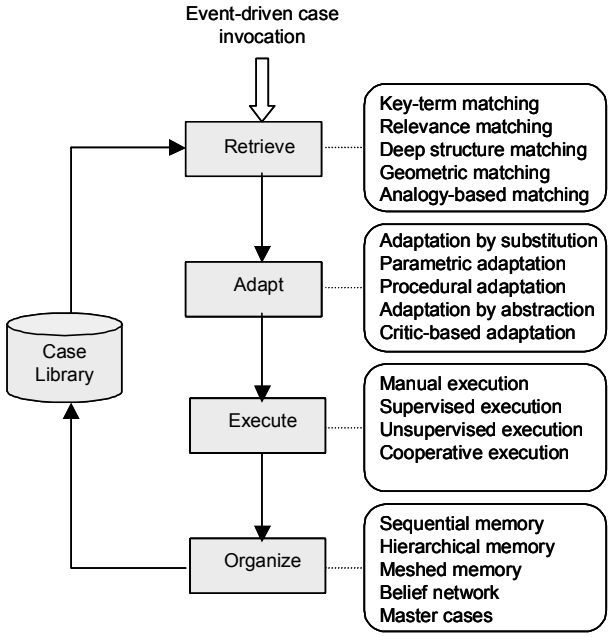

Fig. 3. The case-based reasoning architecture

Importantly, the results of the execution are recorded in the case and the case is entered back into the case library. The design options for the structure of the case library are shown in Fig. 3. In most CBR systems, the case library is structured as a simple sequential list, much like a stack of paper forms. Of course, decision-makers do not structure their problem-solving history in this way. There have been several proposals for more complex memory structures in the literature. An interesting proposal is the concept of a

master case. A master case is one in which all the problem-solving experiences with a particular, well-defined situation are subsumed in one case. This is in contrast with the sequential memory in which each problem-solving experience is confined to a unique case.

Consider a simple abstract example that captures the essence of the CBR approach to cognition. Suppose a retrieved case holds a particular decision for a problem where the decision is based on the value of a variable *x* in some event message:

Retrieved Case
Given situation *S* and parameter *x*, then
Perform action *A*(*x*)
Perform action *B*(*x*)
Make decision $d = C(A(x), B(x))$

Here, *A*, *B*, and *C* may be functions that take a numeric parameter *x* or they may be inferences such as those in classic expert systems that take a symbolic *x* as a known fact. In practice, the user might find that the decision is inadequate because an additional parameter *y* appears that renders the decision unworkable. Further, parameter *x* in the input case that represents the current situation might be some new value of *x*, say *x'*. The user might adapt the plan using parameterized and critic-based adaptation as follows:

Retrieved Case after Adaptation
Given situation *S* parameter x' then
Perform action *A*(x')
If only parameter x' exists then Perform $z = B(x')$
Else if both x' and *y* then Perform action $z = B(x', y)$
Make decision $d = C(A(x'), z)$

Note that the adapted case that is organized in the case library will cover future problem-solving situations in which only *x* is available and in which both *x* and *y* are available. Also, it is expected that further experiences with situation *S* will enhance the knowledge required to perform tasks similar to *S* in future situations. In this way, the system's knowledge is improved with experience. This example shows conceptually four features: First, it shows how the system exhibits some degree of learning with use. Second, it shows how alternative interpretations can be ranked with certainty factors based on the available information. The interpretation produced when both *x* and *y* are available would have higher rank than when only *x* is available, all else being equal. Third, it shows how the system may uncover impediments or opportunities. The case may be retrieved when only *x* is available, whereupon the system advises the operator regarding the utility of *y* or seeks out *y* automatically.

Finally, the example suggests a way to identify bottlenecks and quality-of-service problems in information flow, or meta-fusion functions. Consider again the action *B* and suppose the action is a query to an external database for additional information. The design could include a response time metric *RB* whose value reflects the time that elapses from the execution of *B* to the time that the value of *z* is returned. Alternatively, *RB1* could reflect the time that elapses from the directive to execute *B* to the actual execution of *B*, and *RB2* could reflect the time from the execution of *B* to the return of value *z*. One may design into the system a logger that logs response times of each step of the decision making process into a historical database. Subsequently, after some period of time, the database could be analyzed with OLAP techniques or data mining algorithms in order to discover the actions that are high time-consumers.

A final consideration: The usual approach towards populating a case library is to cast a small set of classic domain-specific situations in the form of cases. In the literature, such a small set of cases is called a seed case library. A seed case library is used to jump-start the growth and fine-tuning of a case library.

A commercial CBR system that embodies the concepts in this section is described in [13]. The system operates in the domain of fault management for large heterogeneous communications networks, where it receives events and alarms from multiple network management platforms.

4 Information fusion via real-time event correlation

4.1 Event correlation

Modern sensor systems and information sources may produce a very large number of events and data. This leads to serious difficulties in situation management:

- The inability to follow a stream of incoming events: events may pass unnoticed or be noticed too late
- The incorrect interpretation of events: decision-making is based on a single event rather than on a macroscopic, generalized event level
- The incorrect evaluation of events: management staff concentrate on less important events.

Event correlation is one of the key techniques for managing high volumes of event messages and to recognize complex event patterns

We define the task of event correlation as a conceptual interpretation procedure in the sense that a new meaning is assigned to a set of events that happen within a predefined time interval. The procedure could range from trivial event compression to complex pattern-matching. A typical event correlation involves dynamic pattern matching over a stream of events. The correlation pattern may include relations among objects in a situation, diagnostic test data, and data from external databases.

Situations unfold in discrete space and time. The time model used here is discrete time with two modifications: point time and interval time. In point time, the events take place at time moments represented as integers in a predefined time scale. For example, the use of Universal Time requires that date/time stamps attached to the event message be reduced to a numeric value in seconds or minutes. Point time is applied to most actions such as user commands. In interval time, events are described by two time moments: the time of origination and the time of termination. Events corresponding to faults, changes in system behavior, or changes in the state of sensor or other control equipment are usually described in interval time.

Depending on the nature of the operations performed on events, event correlation functional types include:

1. [a, a,…, a] → [a]	Compression
2. [a, p(a) ≤ H] → [nil]	Filtering
3. [a, C] → [nil]	Suppression
4. [n × a] → [b]	Counting
5. [n × a, p(a)] → [a, p′(a), p′ ≥ p]	Escalation
6. [a, subset(a, b)] → [b]	Generalization
7. [a, subset(b, a)] → [b]	Specialization
8. [a T b] → [c]	Temporality
9. [a, b,…,T, and, or, not] → [c]	Logic/Temporality

Event compression (1) is the task of reducing multiple occurrences of identical events into a single representative of the events. The number of occurrences of the event is not taken into account. The meaning of the compression correlation is almost identical to the single event a, except that additional contextual information is assigned to the event to indicate that this event happened more than once.

Event filtering (2) is the most widely used operation to reduce the number of events presented to the operator. If some parameter *p(a)* of event *a,* e.g., priority, type, location, time stamp, etc., does not fall into the set of predefined legitimate values H, then event *a* is simply discarded or sent into a log file. The decision to filter event *a* out or not is based solely on the specific characteristics of event *a.* In more sophisticated cases, set H could be dynamic and depend on user-specified criteria or criteria calculated by the system.

Event suppression (3) is a context-sensitive process in which event *a* is temporarily inhibited depending on the dynamic operational context *C* of the operations management process. The context *C* is determined by the presence of other event(s), available resources, management priorities, or other external requirements. A subsequent change in the operational context could lead to delivery of the suppressed event. Temporary suppression of multiple events and control of the order of their exhibition is a basis for dynamically focusing the monitoring of the operations management process.

Another type of correlation (4) results from counting and thresholding the number of repeated arrivals of identical events. Event escalation (5) assigns a higher value to some parameter *p(a)* of event *a,* usually the severity, depending on the operational context, e.g., the number of occurrences of the event. Event generalization (6) is a correlation in which event *a* is replaced by its super class *b.* Event generalization has high utility for situation management. It allows one to deviate from a low-level perspective of events and view situations from a higher level. Event specialization (7) is an opposite procedure to event generalization. It substitutes an event with a more specific subclass of this event.

Correlation type (8) uses temporal relation T between events *a* and *b* to correlate depending on the order and time of their arrival. Event clustering (9) allows the creation of complex correlation patterns using Boolean operators over conditional (predicate) terms. The terms in the pattern could be primary events or the higher-level events generated by the correlation process. In this paper we are concerned primarily with the logic and temporality aspects of event correlation.

4.2 Temporality of event correlation

Formally, an event is a pair [message, time quantifier] in which the message describes the content of the event and the time quantifier is a moment in point time or a time interval reflecting the duration of the event.

Each event correlation process has an assigned correlation time window, i.e., a maximum time interval during which the component events should happen. The correlation process will be started at the time of arrival of the first component event *a* and stopped as the last component event *c* arrives. Like any other event, each correlated event has its time of origination, time of termination, and lifespan. By definition, the time of origination of the correlation is equal to the time of origination of the last component event.

Event correlation is a dynamic process in that the arrival of any component event instantiates a new correlation time window for some correlation. This means that the correlation time window slides in time to capture new options to instantiate a correlation. However, if temporal constraints are assigned to the component events, e.g., when event *b* should be always after event *a*, no correlation time window is started when *b* arrives.

The length of the correlation window and the lifespan of an event correlation directly affects the potential of creating correlations. Widening the correlation window and increasing the lifespan increase the chance of creating a correlation. For very fast processes, e.g., a burst of events from multiple signal fusers, the width of the correlation window could be seconds; while for slow processes such as analyzing a trend of failures from an event log file, the correlation window may be several hours or several days long. The same is true for the lifespan: informative events could last several seconds while the lifespan of critical events should be indefinite, i.e., these events always should be cleared by the operator or by the system. The right value for the correlation window and the lifespan emerge from the application of event correlation to specific domains.

Temporal event correlation plays a critical role in cognitive fusion. A cognitive fusion system should be able to reason about the relative and absolute times of occurrences of events, duration of events, duration of absences of events, and sequences of events. The time interval between events can be defined on a quantitative time scale or on a qualitative time scale.

4.3 Example

The following scenario illustrates how event correlation rules and invocation of a situation case could be built in an engine such as described in [9]. Suppose an event of type A issued at time t1 from a some tank labeled as *?tank1*, but during the following 1-minute (60 second) interval an expected event of type *B* was not issued from some tank *?tank2*. It is also noted that tanks *?tank1* and *?tank2* form a unit, where *?tank1* is the leader and tank *?tank2* is the

deputy supporting tank *?tank1*. The prefix '?' refers to a variable.

```
RuleName:  UNIT-SUPPORT-CORRELATION_RULE
Conditions:
   MSG:    EVENT-TYPE-A              ?msg1
           TANK:                     ?tank1
           TIMESTAMP                 ?t1
   Not     MSG: EVENT-TYPE-B         ?msg2
           TANK:                     ?tank2
           TIMESTAMP                 ?t2
   AFTER:  TIMESENT ?t               ?t1 ?t2 60
   REL     SUPPORTED-BY
                LEADER               ?tank1
                DEPUTY               ?tank2
   Actions:
   Assert: UNIT_COTNACT_LOST_CASE
           ATTRIBUTES
                msg1, ?tank1, ?msg2, ?tank2, ?t
```

The events to be correlated, then, are A and not-B. Note that not-B is treated formally as an event. The additional constraints are that (i) a temporal constraint that the event not-B comes 60 seconds later than A; this constraint is implemented using the temporal relation AFTER, and (ii) tanks are in a unit, where the second tank supports the first one; this constraint is implemented using a domain specific relation SUPPORTED_BY.

If the conditions of the rule UNIT_SUPPORT_ CORRELATION_RULE are true, then the request UNIT_COTNACT_LOST_CASE with the attribute values msg1, ?tank1, ?msg2, ?tank2, and ?t is sent to the CBR engine. The CBR engine either accepts the request as is, or adapts an existing case to match the request conditions. A system that embodies the concepts in this section has been tested and fielded in the domain of telecommunications management [12].

5 Cognitive fusion: an integrated system architecture

5.1 Realizing an integrated architecture

At the architecture level the integration of event correlation and CBR presents a number of interesting challenges that have not been addressed in prior systems. These challenges include real-time processing, synchronizing temporal regions of interest, coordinating the semantic representation used by each system, and incorporating the modeling of situations in the integrated EC-CBR system. An early instance of CBR for a real-time application was described by the second author in [14].

In this section we discuss two aspects of integrated cognitive fusion: (i) the aspect of integration on a conceptual level, where the cognitive fusion functionality is achieved by integration of EC and CBR, and (ii) on a system architectual level, where a cognitve fusion service is architected as a distributed system containing multiple integrated component services. As it will be shown in Sec. 5.3, cognitive fusion service is a part of a larger dynamic situation management system.

5.2 EC and CBR integration

As it was discussed in Sec. 2.3 the modeling of dynamic situations is achieved by situation state transitions. Such transitions may involve different degrees of complexity and are the result of event-situation interactions between the real-time EC and the CBR processes. The main interaction scenarios that these processes are engaged in are as follows:

(i) **Direct situation update scenario**. According to this scenario the updates, including new attribute values, creation/deletion of entities, and establishment/ modification/deletion of inter-entity relations, are performed directly by the CBR process as response to single events passed through the EC process without involving event correlation. The updates may also be performed as requests from management commands, actions from outside systems, or driven by lifecyle management processes such as automatic creation/ modification/deletion of entities according to a schedule or depending on the consumption of internal resources that the entity possesses.

(ii) **Correlation-situation scenario.** This scenario, which corresponds to the fusion-driven transition in Sec. 2.3, is a mode of interplay between the EC and CBR processes, where the EC process generates a hypothesis regarding a new situation, and the CBR process attempts to corroborate the hypothesis by selecting a case, adapting it to the current conditions, and building a new overall situation.

(iii) **Situation prediction scenario.** According to this scenario, the CBR process determines potential future situations not as a direct event coming from the EC process, but rather observing the historic similarity between situations and using the analogical reasoning power of CBR to predict potential continuations of the current situation. This scenario corresponds to the CBR-driven transitions discussed in Sec. 2.3, and is used for threat analysis and for prediction of other undesirable situation transitions.

(iv) **CBR feedback scenario**. This scenario encompasses the contextual information that could be fed back to EC from CBR in case of incomplete or conflicting information that the EC has. In principle, from the correlation viewpoint, this feedback information could be considered just as another "outside" event. An extension of this CBR function could be a CBR driven situation simulation environment for training, testing, or other purposes

Fig. 4 shows an integrated architecture with an event correlation engine at the front end and a CBR engine on the back end. Based on the scenarios discussed above, the design of the conceptual architecture is straightforward. Fig. 4 emphasizes the flow of information in both directions between the correlation engine and the CBR engine. The events that are issued from the correlation

engine are used to invoke cases, or situation templates, that serve as interpretations of multiple events.

In the reverse direction, as described in the CBR feedback scenario, a case might suggest additional contextual information to the EC process, which, if it were available, would strengthen the hypothesis.

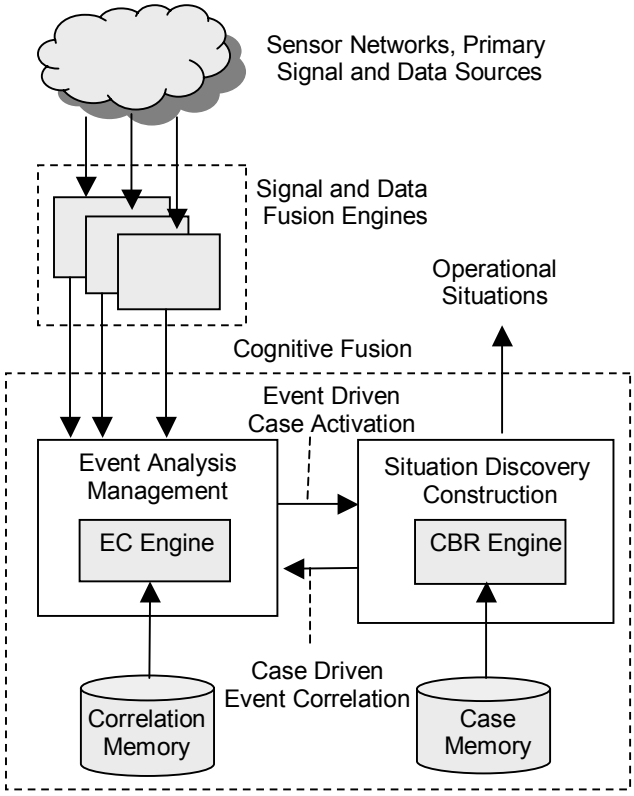

Fig.4. EC and CBR integration architecture

5.3 Distributed services

The foundation for implementation of the cognitive fusion system is a distributed situation management architecture (see Fig. 5). The use of standard services or components with well-defined functionality and standard inter-component communication protocols allows the building of open, scalable, and customizable systems. The encapsulation of the idiosyncrasies of components and the addition, replication, and replacement of components provide an effective environment for developing multi-paradigm, fault-tolerant, and high-performance systems. Various technologies can be used for building the infrastructure of distributed systems, e.g., CORBA, RMI, and Jini [15-17].

We identify Core System Services such as Naming, Directory, Time, Subscription, and Logging services, which are used as the major building blocks to build the Applications Services. There are four types of real-time application services, the Signal, Data, Event Correlation and the CBR services, where the last two constitute the Cognitive Fusion Services.

The architecture in Fig. 5 depicts Topology, Data, Ontology, and Knowledge Services. In addition, the architecture includes four more types of services: (i) the Event Mediation, which performs the connectivity and protocol conversion functions so that sensor and intelligence data can reach the Signal, Data, and Cognitive Fusion services, (ii) Data Adaptation to perform data and knowledge translation functions, (iii) Data Security, and (iv) Event Notification.

Different instances of the services can be used as long as they satisfy overall functional and data semantic constraints. For performance or functional reasons, multiple processes of the same service could be launched. For example, a hierarchy of Event Correlation Services could be created. This hierarchy could be used to implement a multilevel Cognitive Fusion paradigm, e.g., to implement local and global correlation functions.

The Event Notification Service enables event-passing interfaces between distributed objects—the producers and consumers of events. The interfaces are mediated via event channels that allow decoupling of producers and consumers in the sense that they possess no knowledge about each other. The CORBA standard for the Notification Service, OMG's COSNotification Service, defines several important features of the Notification Service, including asynchrony, event subscription, multicast event routing, event filtering, quality of service, and structured events. The output of one channel can be chained to the inputs of another channel to create event notification chains, for example from a Sensor Network to a Signal Fusion Service, then to a Data Fusion Service, to a Cognitive Fusion Service and finally, to a Presentation service. Each of the nodes in a notification chain may cache events, take actions, perform some transformation on the events, and forward them along the chain.

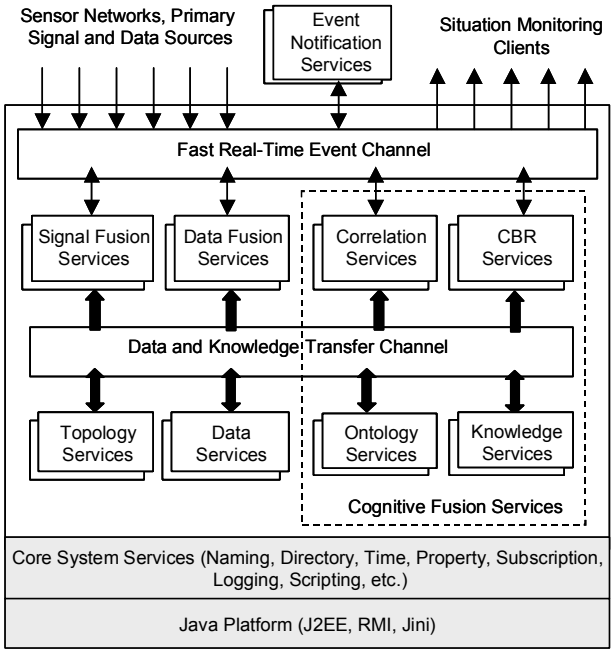

Fig. 5. Distributed situation management architecture

Extendable Markup Language (XML) is used for transporting data and knowledge between distributed components. XML represents arbitrary semantics as strings. It is not necessary to predefine the contents of

these strings for the sake of the transport medium, making XML ideal for transferring data of arbitrary semantics over CORBA. While the distributed systems will define their framework in the CORBA Interface Definition Language (IDL), they will define much of the data semantics in XML. This approach allows components of the system to be decoupled in order to support a consistent knowledge and data transport mechanism.

6 Conclusions and further work

In this paper we propose an integrated approach to *cognitive fusion*, which we define as a process of multi-source data fusion where qualitatively new meaning is assigned to the fused data using cognitive processing techniques. We describe a realization of a cognitive fusion architecture where the two component technologies are real-time EC and CBR. The former is used for reasoning about events from the perspective of time, while the latter is used to model situations. A set of correlated events may trigger the invocation of a case, where a case adds further meaning to the set of events and infers a possible situation. Further, the case may (i) point out additional information that is needed in order to strengthen the belief that a specific situation is present and (ii) offer advice and plans for taking further actions with a desired outcome in mind.

A simplifying assumption of the paper is that signal and data-level fusion have already taken place and have resulted in specifications of discrete events. This task itself is not trivial, and hard challenges and issues are expected to surface when the assumption is lifted.

The integrated model of EC and CBR described here addresses several significant challenges related to how knowledge used by each system is shared, situation transitions are decided, and temporal behavior is coordinated. At the architecture level, the use of CBR in real-time has seen limited use (a notable exception is [14]). We describe at a high-level an approach for using CBR in time-dependent dynamic situation analysis.

The cognitive fusion paradigm opens many interesting research issues, such as improving the predictive component of CBR, incorporating assumption-based reasoning to use cognitive fusion in situations with incomplete information, and learning new situations.

From the practical system development perspective, further work includes: (i) establishing a use case model and accompanying prototype GUIs for actual field operation in some domain, (ii) verifying and validating the integration of the two technologies in some domain, and (iii) providing walk-throughs of system operation.

Acknowledgements

The authors acknowledge Colonel Ed Sherman (Ret) of the US Army for useful discussions of cognitive fusion, battlefield management, and situation awareness from the perspective of the US DoD.

References

[1] Ikuo Tahara, editor. *Cognitive Information Processing*. IOS Press, Amsterdam, The Netherlands, 1994.

[2] Allen Waxman, David Fay and Richard Ivey. Multisensor image Fusion & mining: from neural systems to COTS software. In *Proc. International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 355-362, Cambridge, MA, 30 September-4 October 2003.

[3] Erik.P. Blasch, Scott N. Watamaniuk, and Peter Svenmarck. Cognitive-based fusion using information sets for moving target recognition. In Ivan Kadar, editor, *Proc. Ninth Conf. Signal Processing, Sensor Fusion, and Target Recognition,* SPIE volume 4052, pages 208-217, Orlando: FL, 24-28 2000.

[4] Franklin White. Data fusion sub-panel report, In Proc. *1991 Joint Service Data Fusion Symposium*, volume 1, pages 335-361, 10 October, 1991.

[5] Alan N. Steinberg, Christopher L. Bowman, and Franklin E. White. Revisions to the JDL data fusion model, In Proc. *NATO IRIS Conf.*, Quebec, Canada, October 1998.

[6] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In Donald Michie, editor, *Machine Intelligence 4*, American Elsevier, New York, NY, 1969.

[7] Richard D. Gilson. Introduction to the special issue on situation awareness. *Human Factors*, 37(1): 3-4, 1995.

[8] Fiora Pirri and Raymond Reiter. Some contributions to the situation calculus. *J. ACM*, 46(3): 325-364, 1999.

[9] Hector J. Levesque, Raymond Reiter, et al. GOLOG: a logic programming language for dynamic domains, J. Programming, 31: 59-84, 1997.

[10] Dong Wen-Yu, Xu Ke, and Lin Meng-Xiang. A situation calculus-based approach To model ubiquitous applications. In *arXiv*, Cornell University e-print Archive, 2003.

[11] Chris J. Matheus, Mitch M. Kokar, and Kenneth Baclawski. A core ontology for situation awareness. In *Proc. Sixth Int. Conf. Information Fusion*, pages 545 -552, Cairns, Australia, 8-11 July 2003.

[12] Gabriel Jakobson, Mark Weissman, Leonhar. Brenner, Carol Lafond, and Chris Matheus. GRACE: building next generation event correlation services. In *IEEE Symp. Network Operations and Management,* pages 701-714, Honolulu, HI, 10-14 April 2000.

[13] Lundy Lewis. *Managing Computer Networks: A Case-Based Reasoning Approach*. Artech House Publ., Norwood, MA, 1995.

[14] Lundy Lewis. Method and apparatus for resolving faults in communications networks. US Patent Number 5,666, 481. 9 September 1997.

[15] Jini Network Technology. http://wwws.sun.com/software/jini/

[16] Jon Siegel. *CORBA Fundamentals and Programming*. John Wiley & Sons, 1996.

[17] Troy B. Downing. *Java RMI: Remote Method Invocation*, IDG Books Worldwide, 1998.