Published in the Proceedings of the Second International Conference on Collaborative Computing (CollaborateCom 2006), Atlanta, Georgia, USA, Novermber 2006

Awareness-based Collaboration Driving Process-based Coordination

Dimitrios Georgakopoulos, Marian Nodine, Donald Baker, Andrzej Cichocki
Telcordia Technologies, 106 E. Sixth Street, Suite 415
Austin, Texas 78701, USA
{dimitris, nodine, dbaker, andrzej}@research.telcordia.com

***Abstract*—Awareness-Enabled Coordination (AEC) is a platform designed to address the problem of scaling collaboration to large multi-organizational teams. Such collaboration is inhibited by the complexity in multi-organizational environments and lack of efficiency in achieving team objectives. AEC provides a contextualization mechanism that deals with such complex, real world environments where teams involve humans, tools, software services, and agents that come from different organizations, are subject to multiple jurisdictions, and provide diverse expertise. To provide efficiency in achieving team objectives, AEC provides situation- and project-related awareness, as well as process-based coordination and automation. We describe the AEC architecture and discuss AEC models and mechanisms for computing awareness and coordinating action. We use examples from the homeland security domain to illustrate these AEC technical capabilities and their benefits.**

***Keywords*—awareness; process; coordination; collaboration**

I. INTRODUCTION

Many of the technologies and software products that have been developed to support virtual team collaboration [15][27][12][13] have problems with scaling to large multi-organizational teams. This is due to the complexity in multi-organizational environments and lack of efficiency in achieving team objectives. While some existing technologies support many users, they provide only limited help for users and virtual teams to deal with the complexity of the environment in which they operate. Additionally, they do not provide models and corresponding mechanisms geared towards significantly increasing efficiency in achieving team objectives.

Awareness-Enabled Coordination (AEC) is a platform designed to support effective collaboration of large multi-organizational teams, possibly operating in dynamically changing situations (e.g., disaster response). Collaborating team members may be geographically distributed, be employed by or be serving multiple agencies or organizations, and may have different processes and resources, even for doing similar activities; these factors conspire to make the collaboration environment very complex. Unlike other existing technologies for supporting collaboration, AEC supports the ability of its users to deal with the complexity of their environments; specifically, the events, processes, and resources that often arise from various contexts that reflect different organizations, jurisdictions, teams, and activities. AEC deals with such complexities by providing capabilities for modeling such contexts, as well as *contextualization* mechanisms for automatically mapping events, activities and resources from one context to another, and combining these in composite events and processes across virtually any network of contexts that reflects the real world environments being supported by AEC.

This material is partially based upon work supported by Air Force Research Laboratory, Information Directorate, Rome, NY under Contract No. F30602-03-C-0006.

AEC's *context management* capability provides models, tools and repositories that organizations, jurisdictions, teams, and persons can use to model (and store for reference) the types of events, processes, and resources that are of interest within their own scope, as well as their relationships with other contexts. Typically, the modeling of organization, jurisdiction, and team contexts is accomplished by experts within each of these contexts. The personal context of each user is maintained by its owner. The required initial effort for context modeling is worthwhile, since it enables AEC to automate the *contextualization* of events, resources, and activities.

Events are evaluated, activities are performed, and resources are utilized under the aegis of one or more contexts, with which they are related, either directly or transitively. When a user declares his/her need to be alerted of a complex event or his/her intent perform an activity, *dynamic contextualization* utilizes this related context information to automatically determine possible ways for mapping the related events that originate in other contexts to one or more (possible complex) events in the user's context or suggesting ways for doing the intended activity, the resources it may use, and the events that may apply to it. Contextualization effectively reduces the space the user has to consider by helping users focus only on the events, processes, resources, and events that relate to their own context and filtering out those contexts and their contents that do not apply.

To increase team communication and coordination efficiency, AEC takes advantage of contextualization in providing the following synergistic capabilities:

- *Provide situational and team awareness to each team member by utilizing contextualized, possibly complex events*: AEC provides awareness both with respect to the progress of the team towards the completion of an activity, and any situational changes that may impact their work. Awareness is computed using user-defined awareness specifications. Evaluation of awareness specifications involves the automatic monitoring the information in all contexts and other external event sources, analyzing them to detect specified events and event patterns, and delivering such awareness via alerts to the targeted users. Efficient team communication is accomplished via contextualized, possibly complex events indicating potential problems or progress.

- *Provide process-based coordination and automation of team actions by utilizing awareness*: AEC supports the specification/modeling and automation of organizational, team, and information sharing processes. AEC-supported processes are flexible. They accommodate team and individual styles of work ranging from highly structured business processes to dynamically self-organized work. This increases efficiency of team coordination and action.

The remainder of this paper discusses these capabilities, provides examples, and describes related tools provided by AEC. It is organized as follows: Section II provides an overview of AEC context management. Section III describes dynamic event contextualization, while Section IV discusses the specification and computation of awareness. Section V describes process-based coordination and automation. Section VI focuses on the contextualization of activities and processes. Section VII gives a brief overview of the AEC architecture. We describe related work and conclude in Sections VIII and IX, respectively.

II. Context Management

A *context* is a mosaic of information, knowledge, resources, and programs that are gathered together for a particular purpose. Examples of contexts supported by AEC include those that gather the events, process, and resources for organizations, jurisdictions, real-word objects and locations, teams, and individuals. These are the elements each specific context provides to help AEC users achieve the context's purpose. Administrators with appropriate authority and training use AEC context management tools to populate and maintain their contexts, storing their current information and knowledge in AEC repositories, and making it accessible within their own context and to related contexts.

An AEC context consists of a *scope* and a set of *context elements*. Currently, AEC context elements include directly accessible *events*, *resources*, and *methods* that have meaning relative to the context. Context elements may include references to or copies of elements in other contexts. The context scope provides *referential relationships* to other contexts containing relevant events, resources, methods, etc. The intent of the scope is to provide boundaries on the visibility and accessibility of elements in other contexts. A set of contexts that are interconnected with context references forms a *context network*.

Fig. 1 depicts an example of an AEC context network from the homeland security domain, shown as a set of contexts and a

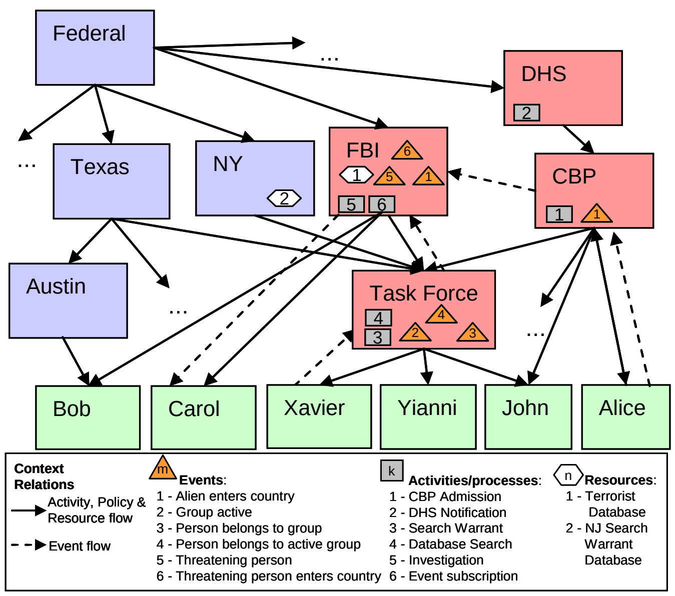

Figure 1. Context network example.

set of relationships. Two types of relationships are shown as arrows in Fig. 1: *event flow*, and *resource and activity, policy and resource flow*. Referential relationships are defined in the opposite direction of the arrows in Fig. 1. For example, the event and resource flow relationship between the contexts of Department of Homeland Security (DHS) and the Customs and Border Protection (CBP) is defined by including DHS in CBP's scope. Event flow is determined by publish/subscribe relationships as shown by the dotted arrows in Fig. 1.

Each organization (e.g., government agency) has its own *organizational context* that typically includes its resources, events, and processes. Large organizations may have a hierarchy of organizational contexts related via (possibly typed) referential relationships that mirror its organizational structure. For example, in Fig. 1 the CBP organization is a part of the DHS. All DHS processes apply to CBP, and DHS resources may be usable by the CBP (unless they are access restricted). This is reflected in the relationship between the two contexts.

Jurisdictional contexts (e.g., the Federal, Texas, NJ, and Austin contexts in Fig. 1) maintain the laws, events, processes, and resources (e.g., the database of people entering the US, and the roles of judges for issuing search warrants), and other information for each jurisdiction. The Federal context is typically at the top of the government's jurisdictional context network. Changes in both jurisdictional and organizational contexts are relatively infrequent. As with all contexts, their resources, activities, and policies flow to other contexts within their scopes, as needed.

Contexts for real-word objects and locations may include resources (i.e. actors or other objects of interest) and their attributes; state changes of the resources; time intervals of state changes; spatial coordinates of the entities (if any); and relations of these entities to other information in such contexts. For example, the context of the disaster response organization for the city of Austin includes available resources (e.g., police cars, fire tracks, people, and teams of these) and their up-to date spatial coordinates, event types for monitoring their state changes, as well processes for deploying such resources for each kind of anticipated emergency.

The contexts at the bottom of Fig. 1 are personal contexts for the individuals that belong in different organizations. These individuals may be subject to multiple jurisdictions. For example, Carol's context in Fig. 1 is subject to the processes and events in FBI, Texas, and New Jersey. This is indicated by the "event and resource flow" relationship between Carol's context and the FBI, Texas, and NJ contexts.

Increasingly complex context interrelationships apply in situations where multi-organizational teams (such as a task forces and emergency preparedness teams) include members that operate under multiple jurisdictions and organizational processes. The Task Force in Fig. 1 is an example of such a multi-organizational team. For example, suppose that the Joint Team consists of FBI and CBP agents and is has been formed to investigate a suspected terrorist that enters the US. The scope of the Task Force context includes the FBI and CBP contexts (as well as DHS via CBP). The Task Force members (i.e., Xavier, Yianni and John) are subject to the processes and can use the resources specified in the new team context, their organizational contexts (i.e., FBI and CBP), as well as the jurisdictional contexts of the states they operate. Processes, resources, and events in these contexts that are relevant to the team-related activities performed by the Task Force members will be determined via dynamic contextualization.

III. DYNAMICALLY CONTEXTUALIZING EVENTS

To describe event contextualization, we first need to define *events*. In AEC, *events* are packets of information describing an occurrence. Packetizing information in an event allows it to have a lifetime separate of the occurrence it describes. Therefore, events allow reasoning about the event occurrence to be disconnected in time and place from the occurrence itself, a requirement in situations where event sources are distributed. A *primitive event* describes a real-world occurrence that can be either detected directly by AEC (e.g., completion of an activity or a change in the status a resource managed by AEC) or it is detected by an external event source that is monitored by AEC (e.g., a sensor, a person). Therefore, resources, programs, and human activities are sources of primitive events. A *composite event* describes a constellation of related events (either primitive or composite) that has meaning as a complex occurrence (e.g. a project completing with its deadline and budget constraints). An *alert* is a composite event that is delivered to a user. Such alerts are the basis of awareness in AEC.

The first step in making these primitive events understandable to the end users is to relate them to one or more *contexts*. We call this event transformation *contextualization*. Another issue with external event sources such as sensors or people is that the same real-world event might be "seen" by multiple sources with each source providing partial, but overlapping information about the occurrence. Such redundant

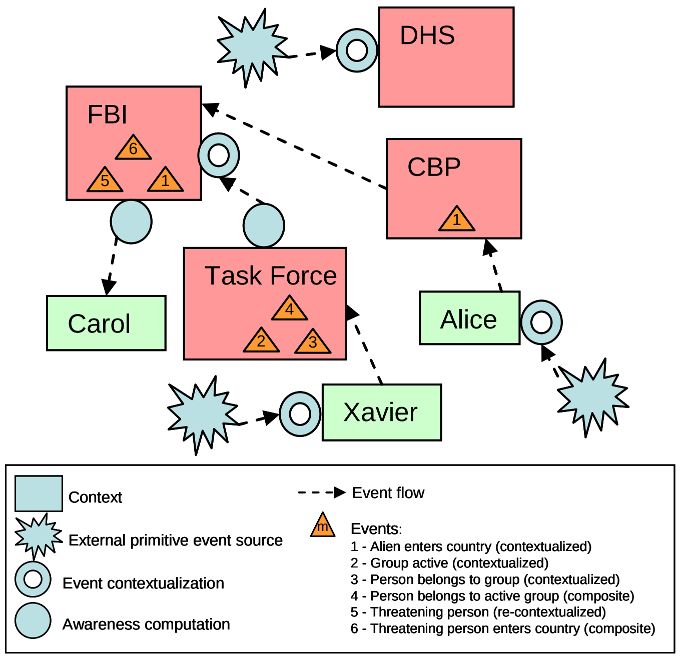
DHS
FBI
6
5
1
CBP
1
Task Force
4
2
3
Carol
Alice
Xavier
Context
External primitive event source
Event contextualization
Awareness computation
Event flow
m
Events:
1 - Alien enters country (contextualized)
2 - Group active (contextualized)
3 - Person belongs to group (contextualized)
4 - Person belongs to active group (composite)
5 - Threatening person (re-contextualized)
6 - Threatening person enters country (composite)

Figure 2. Event contextualization and awareness computation in AEC contexts.

(possibly heterogeneous) information must be fused. To provide near-real time awareness, contextualization and fusion must be performed in a data-driven fashion as new primitive events arrive. Finally, because some event sources may need unknown/unanticipated time to produce events, contextualization and fusion of primitive events must not embody expectations concerning delays, such as computational time windows.

To addresses these needs, AEC provides a suite of *Event Contextualization* (EC) and *Awareness Computation* (AC) capabilities. Fig. 2 uses a subset of the contexts we introduced in Fig. 1 to illustrate the utility of these capabilities. In the following paragraphs, we focus on EC. AC is described further in Section IV.

The purpose of Event Contextualization in AEC is to automate the transformation of primitive events that are gathered from external event sources to one or more AEC contexts. For example, event contextualization is applied in our example in Fig. 2 to automatically transform and fuse event reported by Xavier and Alice to their personal contexts in AEC. Event contextualization is also applied to automatically transform and fuse events detected in one AEC context to events of different types in other AEC context that subscribe to the source context (e.g., subscriptions are illustrated as event flow relationships in Fig. 2). For example, events detected in the DHS and CBP contexts in Fig. 2 are contextualized (i.e., transformed and fused) into events of interest to the FBI context. Therefore, a *contextualized event* is an event (re)cast in terms of the concepts represented in a target context. Such an event may have parameters that identify the actors, their activities, and the space and time they occur.

Upon receipt of an event, an Event Contextualization capability must perform the following activities:

1. Contextualize the event by correlating event parameters and event source metadata (e.g. the AEC resource, user, process, external event source, and location) with the information of related contexts. (e.g., the organizational and personal contexts).
2. Incrementally fuse the primitive event with information already present in the context for the specific entities related to the event. This results in an update to the context (e.g., when multiple sources report the occurrence of an event, the information carried by the events they produce is fused into a single event in the context).
3. Incrementally publish the resulting event(s) to the subscribing contexts.

Event contextualization is unnecessary and is not performed in situations were the source and subscribing contexts use the same event types (e.g., in Fig. 2 the CBP context subscribes to the same events as Alice's personal context).

AEC's Awareness Computation capabilities are applied only on contextualized events as we describe next.

IV. Computing Awareness

We define awareness as the stream of events that carry highly relevant information to a specific user role and situation. Because a human's attention is a finite resource that must be optimized, awareness events must be digested and delivered to exactly the users who need them via alerts. The information in the awareness event must be related to the context that gave rise to it. Note that, if awareness events provide less information or they are targeted improperly, users will act inappropriately or be less effective. Users receiving too many or uninteresting events must deal with an information overload that adds to their work and masks important information.

To compute awareness, AEC provides an *Awareness Computation* (AC) capability that consumes contextualized events in each context and detect complex events of interest to users or other subscribing contexts. When it detects such complex events, AC embodies them in *alerts* targeted to users playing context specific roles.

Our approach in AEC is that the awareness that users receive must be explicitly specified in advance. AEC detects complex events and generates the corresponding alert and task requests based on user-authored *awareness specifications*. Awareness specifications are comprised of context-specific interconnected computational units called *event operators*.

The inputs and outputs of event operators are streams of contextualized events, either primitive or composite. Connections between operators are only allowed between semantically compatible event types, as described by the event ontology (AEC's event ontology is represented in full OWL [18]). This restriction helps ensure compatibility between producing and consuming event operators, but it also ensures semantic compatibility, so that the overall awareness

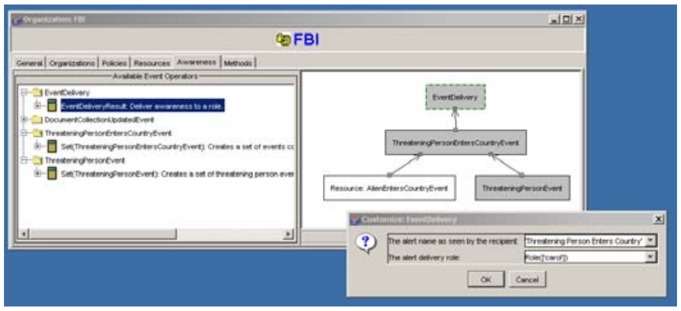
Organization: FBI
FBI
General
Organizations
Policies
Resources
Awareness
Methods
Available Event Operators
EventDelivery
EventDeliveryResult: Deliver awareness to a role.
DocumentCollectionUpdatedEvent
ThreateningPersonEntersCountryEvent
Set(ThreateningPersonEntersCountryEvent): Creates a set of events cc
ThreateningPersonEvent
Set(ThreateningPersonEvent): Creates a set of threatening person ever
EventDelivery
ThreateningPersonEntersCountryEvent
Resource: AlienEntersCountryEvent
ThreateningPersonEvent
Customize: EventDelivery
OK
Cancel
The alert name as seen by the recipient: 'Threatening Person Enters Country'
The alert delivery role: Role('carol')

Figure 3. An awareness specification in AEC.

specification computes complex events/alerts relative to the author's understanding.

Authorized AEC users utilize a graphical editor to author awareness specifications for each context, as needed. To author an awareness specification a user would create event operators from a palette of operator types, interconnect the event operators' input and output event streams, and customize the operators' computational behavior via dialog boxes. AEC provides a suite of generic operators that can be customized for each context. Operator customization in AEC may be performed via dialog boxes or via a programming language.

Fig. 3 shows the AEC awareness editor being used to specify a *Threatening Person Enters Country* event in the FBI context. The FBI context, its event sources (i.e., CBP, and Task Force contexts) and subscriber (Carol's personal context) are illustrated in Fig. 2. To define the awareness specification for detecting *Threatening Person Enters Country* events, AEC users drag operator instances into the specification from a palette of operator types, as shown on the left side of Fig. 3. The graph diagram for this specification is depicted in the right side of Fig. 3 and involves four operators. The operator *Threatening Person* detects events of type *Person belongs to active group* that are computed in the Task Force Context. Such events are contextualized as a *Threatening Person* event in the FBI context. The *Alien Enters Country* operator simply detects events of the same time that are detected in the CBP context (since both the FBI and CBP context share this event type, *Alien Enters Country* events do not need to be contextualized in the FBI context. The *Threatening Person Enters Country* operator continuously joins the event streams for *Threatening Person* and *Alien Enters Country* and emits events of type *Threatening Person Enters Country* whenever there is a match between threatening people and aliens that just entered the country. The root operator in Fig. 3 sends alerts to the subscribing party (i.e., in Fig. 2 this is Carol) via her personal context. Fig. 2 does not illustrate the contextualization of *Threatening Person Enters Country* events in Carol's context.

AEC operator types can be categorized in broad functional categories:

- set manipulations – operators that perform set operations over the events on their input streams;
- temporal operators – combining constituent events into temporal sequences;

- joining – combining related events from multiple input streams into composite events on the output stream;
- filtering – culling of uninteresting events from the input in the output;
- grouping and aggregation – grouping of multiple input events from a single stream into aggregated output events; and
- alert operators – delivery of composite events to context-specific roles.

With proper training, AEC users can extend the palette of event operators with customized, domain-specific operators. Such operators can be added to an AEC palette at any time. Operator customization is necessary for the following reasons:

- Events in AEC carry information in the form of event parameters concerning the situation described in their source context. Users need to customize operators to pull the most relevant information out of the constituent/input event(s) and summarize such information in the parameters of the computed composite/output event. Information provided on alerts is available at alert delivery time for user inspection. This computation of event parameters by event operators enables AEC to automatically generate the information on an alert that the user is most likely to need.
- Users need to associate each awareness specification to a specific context. In addition, alert operators need to be assigned to a specific role in the operator context.
- To improve ease of use by non-experts, it is desirable to provide operators relevant to domain-specific functions that they can understand easily. For example, the *ThreateningPersonEnters Country* event operator in Fig. 3 has been specifically created for the FBI context. Its advantage is that it has a clear meaning in the FBI context and it requires little or no customization when it is used in this context.

The edges between the operators represent the flow of events from producer to consumer; as shown on the right side of Fig. 3. The leaf operators of an awareness specification derive events from AEC contexts. Interior operators combine one or more such basic events and generate composite events that describe a situation that is more specific than the situations giving rise to the input events. *EventDelivery* operators at the roots of an awareness specification graph act as delivery instructions for their input events. As shown in the dialog box in Fig. 3, the *EventDelivery* operator can be customized to change the title of the alerts and a role of people to whom to direct the alerts. Awareness operators computed output events related to a specific context. For instance, all operators in the awareness specification in Fig. 3 are context-specific, i.e., they compute awareness from resources of the FBI context and deliver it to a role defined in this context.

Awareness is computed incrementally and continuously. Incremental computation enables awareness to be delivered to targeted users in a timely fashion. Alerts are not just generated once, but can evolve over time as relevant information is updated. Awareness specifications can also be edited "on the fly" with the resulting alerts immediately recomputed.

The goal of computing awareness in AEC is to enable effective decision making. When decisions require action, actions typically involve activities and processes that coordinate and automate team work. In the rest of this paper we focus on AEC support for flexible processes that AEC provides as the means for coordinated action.

V. Coordinating and Automating Activities

Coordination and automation of team activities enhances efficiency. Flexibility permits timely action in response to important events that arise while an activities or an entire process is being executed. Such awareness-enabled action typically involves dynamic process refinement and change. AEC provides a *flexible processes model* and corresponding context-based process management mechanisms to enhance the efficiency of coordinating and information sharing among members of multi-organizational teams in a dynamic setting. When processes or parts of processes are well-structured and well-defined, AEC provides the option to automate them, reducing the work load on team members. In the following paragraphs we describe AEC's flexible process model, and describe in more detail its novel capabilities for process-based coordination and automation.

AEC's flexible process model permits interleaved definition, refinement, and execution of activities and processes. A process activity in AEC is a collection of child activities, possibly constrained by dependencies on their execution. A child activity may be intended to be done by humans, be a program or service that is accessed directly (either with the help of a human or in an automated fashion), or in turn be a nested processes. Semantic activity types, called *activity intents*, are defined in AEC's *activity ontology*, whose purpose is to provide a common semantic type system that allows AEC users to indicate their intent when they start a new activity (e.g., by selecting the appropriate semantic activity type in the ontology). Activity intents do not define how an activity is to be done. Thus, the activity's *method* must be defined before activity becomes concrete (i.e., executable by AEC). Thus, a single intent can be associated with multiple methods that satisfy that intent.

In addition to dynamically refining a process activity starting from the intent, AEC supplies *method catalogue* for selecting a method to satisfy the intent of an activity. Each context has an indexed catalogue of suggested (or required) specifications for methods related to specific intents. The user can access the method catalogue from the activity's context based on the intent of the activity, and choose the appropriate entry in the catalogue to use as a basis for his/her own activity. The user may use the selected method as a starting point for further refinement, or may be required to follow a method strictly (for instance, if it is a traditional business process).

Child activities of an AEC process may be constrained in terms of when they can be executed. Like in many traditional process models [29][5][12][8], AEC's process model supports control flow dependencies that order the execution of activities, forcing one activity to precede the other. Resource selection dependencies define the resource types required by each activity in AEC. Resource flow dependencies in AEC are constraints in the flow of resources to or from the context of each activity. If a process has control flow dependencies between all its child activities and methods for all activity intents, we call it a (fully) structured process. Partially structured processes only have control flow dependencies between some child activities, while the child activities of unstructured processes have no control dependencies (i.e., there basically set of activities and of these activity can be performed at any time). Both partially structured and unstructured processes can have activity intents that must be associated with methods by end users before their execution. We use the term *predefined* to refer to activities or processes that have all their methods specified, control flow, resource selection, and resource flow dependencies defined before they are executed. We refer to activities and processes being defined (e.g., by adding a method, changing control and resource flow or adding a resource selection) after the execution of their parent process starts as *dynamically refined*.

AEC's *flexible process model* supports a wide variety of process-based coordination styles ranging from fully *structured* to *unstructured* processes. Furthermore, the AEC flexible process model permits *dynamic refinement and change* of any process during its execution. For example, just as in many other process management systems (e.g., workflow systems and EAI integration platforms [8][3][12]) AEC supports the specification and automation of *business processes* for organizations, jurisdictions, and teams. Using the terms we defined above, business processes are predefined and structured AEC processes that apply to organizational and jurisdictional AEC contexts. Specified business processes can be analyzed and measured to assess and improve their efficiency. Process automation can drastically reduce overhead and cost for assigning tasks to people, coordinating activities, tracking progress towards achieving goals, and maintaining accountability information. AEC provides all these benefits of business process management; however, since we follow well-understood methodologies, we will not discuss these capabilities further in this paper. These automation capabilities give AEC a distinct efficiency advantage over the ad hoc coordination advocated by many groupware (e.g., as in [15].)

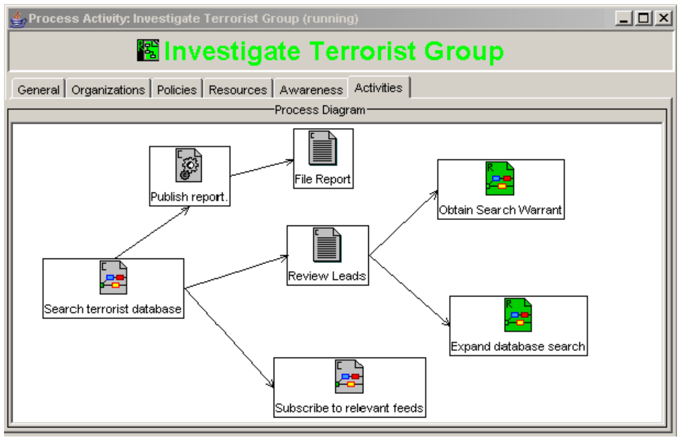

Figure 4. A running AEC process.

Fig. 4 shows a partially completed process for investigating a terrorist group for which FBI agent Carol we introduced in Fig. 2 is a participant. A number of activities have completed, including an initial database search and reviewing the leads that turned up from that search. These activities are performed in the FBI and the Task Force contexts, respectively, that are depicted in Fig. 2. Based on the leads, Carol decided to expand the database search and to obtain a search warrant that would aid in the gathering of information as to the current whereabouts of a particular suspected terrorist. Both of these activities are ongoing as indicated by the "R" for running in the top left corner of the activities' icons. Upon receipt of the alert concerning a potential terrorist entering the country, Carol decides to change the direction of the investigation process. She knows that the search warrant will not be needed as the alert provides her that information. She terminates the activity to obtain a search warrant and creates and starts a new activity to obtain a wiretap on the phone in his known location. The revised process is shown in Fig. 5.

Dynamic refinement and change during process execution permits AEC process execution to start even if a process is only partially defined. The process may be further refined as progress is made towards accomplishing its intent. For example, refinement may occur when decisions concerning the method are made during execution, when a resource is assigned to a child activity immediately before it is executed, or when external events require abandoning planned activities and initiating new unplanned activities in response. For processes, dynamic process refinement modifies the specification of the child activities and the control flow dependencies between them to make the activity more concrete. To accommodate such refinement, an AEC process can include activities that are only specified at the level of what the activity intends to accomplish. (As we discussed earlier activity intents are defined according to a domain-specific activity ontology in AEC). As AEC users obtain more information concerning the details of what needs to be done, the activity may be refined until it is fully-specified.

AEC automates processes to provide further coordination efficiency. Flexible process execution in AEC is performed by AEC's coordination capability, which is process engine functionality that is distributed in all contexts maintained by

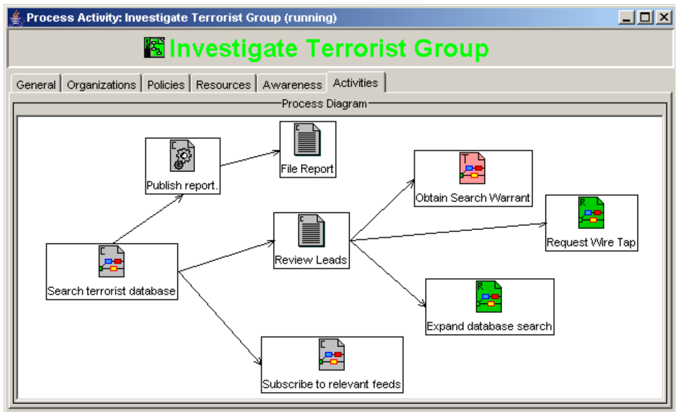

Figure 5. Dynamic process change in AEC.

AEC. When a flexible process starts in a context C, the AEC process engine in C enables the execution of each of its child activities once the following conditions are met on that activity:

- The child activity is defined well enough that execution can begin. Otherwise, the user must refine the activity until it reaches the point where the execution can begin. This may involve defining a process or selecting a preexisting method (e.g., a one that is available in an organizational context).
- The child activity has access to all of the resources on which it depends. Otherwise, the user whose role is specified in the activity is asked to select and bind resources in its context to resources in its environment. The selection of available resources is determined via dynamic contextualization.
- All of the activities' incoming control flow dependencies are enabled.

Once a child activity is ready for execution, it can either run automatically (if it is flagged as *automatic*), or be started by the intervention of some responsible team member. The process may be monitored by any team member, but a specific responsible party (this is a specified activity *role*) is given the task of dealing with any issues during process execution. AEC's provides tools (including the process editing tool illustrated in Fig. 5) for defining, refining, and monitoring flexible processes. Since AEC activities generate events, AEC's awareness tools can used to monitor more detailed aspects for processes.

VI. Dynamically contextualizing activities

Just like events in AEC, every activity in AEC is performed within the direct scope of one or more contexts and is contextualized by AEC. This is necessary because contexts constrain how the activity may be performed by providing the resources and methods that are relevant to the activity. The determination of the set resources, methods, and events that may apply to the activity at the current time is called *dynamic activity contextualization*.

When a user states his or her intent to perform an activity, AEC's activity contextualization mechanism performs the following steps:

1. Dynamically determines the *environment* of the activity. This involves computing the transitive closure of the contexts that are visible from the context of the activity.
2. Reduces the environment of the activity to the set of resources and methods that are compatible to the activity. That is, AEC suggests methods having a type compatible with the activity type, and resources of a type that can be utilized by the activity type. In addition, AEC highlights events that refer to the activity type. Where alternatives exist, AEC either automatically selects preferred alternatives (where they are well-defined) or allows users to make the choice explicitly.

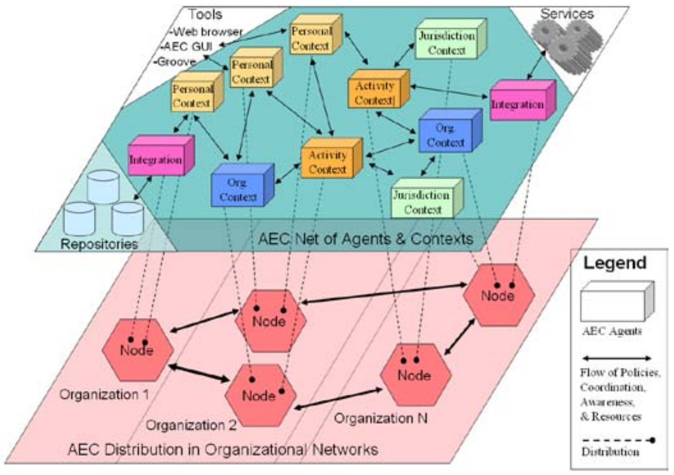

Figure 6. AEC architecture.

In the presence of dynamic change, contextualization involves having each activity constantly track the contexts and context relationships, adapting the activity to relevant changes, establishing access to new elements in other contexts, and dropping references to elements that became inaccessible.

VII. Run time architecture

The AEC platform is comprised of a collection of contexts distributed over a set of nodes that support these contexts. The AEC architecture is depicted in Fig. 6.

Contexts visible to the AEC users are depicted in the upper plane in Fig. 6. As we discussed in earlier sections, examples of AEC contexts include those for organizations, jurisdictions, activities, and individuals.

The arrows between contexts represent the event, resource, and control flow relationships specified between them. The contexts themselves reside on nodes distributed across the network, hosted in different locations and by different organizations or agencies. Each node hosts one or more contexts and provides a set of system services to them, such as messaging, naming, event, etc. Nodes are illustrated in the lower plane in Fig. 6.

VIII. Related work

We described AEC innovations related to context, contextualization, awareness, flexible processes, and dynamic adaptability. In the following paragraphs we contrast our work to existing work in similar areas.

AEC uses the term "context" to refer not only to the immediate context of an activity, but also to the network of jurisdictional, organizational, and other contexts that reflect the environment in which the activity is taking place. Our work concretizes and expands on early work done in the area of organizational contexts and awareness [24]. The domains discussed by Uszok, et al. [26] with respect to KAoS policies are similar to our jurisdictional and organizational contexts. Agent communication contexts in Ricci, et al. [21], when restricted to a single activity of an agent, would maintain similar information to an agent's personal context in AEC.

Also, the information that we place in our context network relates to the types of information you would expect to see at the upper levels of institutions, within the work on representing norms and institutions, e.g. in [9]. However, while related projects compact the contextual policies within the domain or institution, AEC creates a composite context containing related individual, and autonomously-updatable, contexts that can be maintained explicitly by the organizations and jurisdictions whose processes and resources they represent. In addition, AEC provides mechanisms and tools for both automatic and user driven contextualization.

AEC builds on our earlier awareness work [1][2] in process management systems, an advanced process-oriented system. Here, we extend that work to recognize the importance of contextualization as an important aspect of computing awareness. Ultimately, our concept of awareness follows in the spirit of Paul Dourish who advocated the raising the level of abstraction through judicious simplification of the "story a system tells about itself" [10]. We have greatly extended his approach by using situational and contextual relevance to improve the quality of awareness.

The event operators we have advocated for use in the Awareness Computation capability of AEC are based on event processing technologies. Indeed, AEC as a whole can be viewed as an example of an Event Driven Architecture (EDA) [19]. Early event processing systems, such as Snoop [7] developed *event algebra* based models, with generic event operators such a filter, sequence, and count. CEDMOS [6] moved toward self-contained events and the need for computation of event parameters for complex events. Although these systems have explored basic event processing ideas, usability, efficiency, and contextualization were not addressed.

With respect to process-based coordination and automation, many existing workflow systems (e.g., COSA [8], FileNet [12]), Enterprise Integration platforms (e.g., WebLogic Integrator [3], and NetWeaver [23]) as well as standards for process workflow management [29] and process-based web service integration [5] are all geared towards modeling and automating processes that are predefined and fully structured. Therefore, these technologies and standards lack the flexibility of AEC's flexible process model that is necessary to support teamwork in changing environment. The Collaboration Management Infrastructure (CMI) system [14] and it commercial derivative ATLAS [25], as well as others [4][17], have explored relaxing control flow constraints to support some partially structured processes. Other researchers have designed systems such as Caramba [11] that permit either structured or ad hoc activities, or have proposed formal frameworks for dealing with dynamic process change [27][21]. None of these technologies supports contextualization or dynamic change [13].

Existing groupware tools such as Groove [15] typically rely on informal human coordination. When such tools are used for large scale collaboration, the overhead involved in exchanging coordination messages hinders collaboration efficiency. Another problem is that their basic capabilities for contextualizing messages and information do not scale up as well. AEC addresses this issue by scoping activities, events and resources into appropriate contextual settings. Document sharing systems such as Vignette [27] support the sharing of documents and other resources, but provide only token support for coordination.

Finally, although various workflow systems support process monitoring and groupware tools provide limited awareness on the status of shared resources, (e.g., [16]), none of these technologies provides models and mechanisms for customizing situation, and work related awareness to serve the needs of each user. ATLAS [25] and its precursor CMI [14] provide such capabilities but lack contextualization and unrestricted dynamic change.

IX. CONCLUSION

AEC supports the efficient collaboration of large multi-organizational teams. Unlike existing collaboration systems, AEC achieves collaboration efficiency by improving the following collaboration aspects:

1. AEC provides team/project and situational awareness via composite events that combine information from various contexts. Only events that are relevant to the situation and activities of each individual team member are automatically contextualized to his/her/its personal context and delivered automatically via alerts.
2. AEC support awareness-enabled coordination. It facilitates human understanding of dynamic change by harnessing changes to resources and other time-based information into information that is highly relevant to the work and roles of each user. Properly informed users can make better decisions and they are empowered to do so through authoring processes, selecting activities to execute, refining the processes in which they play a part, or changing the structure of resources or even organizations.

These aspects of AEC are in turn supported by provided mechanisms for contextualization and the flow of events, resources, and control. The combination of these capabilities helps users and virtual teams deal with the complexity of their environment, and provide efficiency in achieving team objectives. These capabilities make AEC suitable for supporting large scale collaboration.

The capabilities described in this paper are part of our working AEC research prototype. While AEC has yet to be deployed, we have developed a number of scenarios and demonstrations of AEC's capabilities for our government customer. One such scenario was developed by an external organization comprised of intelligence analysts that captured the extensive, but fictional cross-organizational interactions that might occur in a counter terrorism response. We were able to capture the organizations, events, teams, and processes successfully for this scenario in AEC with relatively small effort. We view this as a small but nontrivial validation of our approach. We are currently looking for early AEC adopters.

References

[1] D. Baker, D. Georgakopoulos, H. Schuster, and A. Cichocki, "Customized Process and Situation Awareness," *International Journal of Cooperative Information Systems*, March 2002.

[2] D. Baker, D. Georgakopoulos, M. Nodine, and A. Cichocki, "From Events to Awareness," Proceedings of *International Workshop on Event-driven Architecture, Processing, and Systems* (EDA-PS'06) ICWS/SSCS 2006, Sept. 2006.

[3] BEA: WebLogic Integrator, http://www.bea.com/, 2006.

[4] D. Bogia, and S. Kaplan, "Flexibility and Control for Dynamic Workflows in the wOrlds Environment," Proceedings of *ACM Conference on Organizational Computing Systems*, 1995.

[5] "Business Process Execution Language for Web Services Version 1.1," http://www-106.ibm.com/developerworks/webservices/library/ws-bpel.

[6] A. Cassandra, D. Baker, and M. Rashid, "CEDMOS, Complex Event Detection and Monitoring System," MCC Technical Report CEDMOS-002-99, Microelectronics and Computer Technology Corporation, 1999.

[7] S. Chakravarthy, "Snoop: An Expressive Event Specification Language for Active Databases," Data and Knowledge Engineering, 14(10), 1994.

[8] COSA Solutions: COSA Product Suite, http://www.cosa.de/, 2003. Damianou, N., N. Dulay, E. Lupu and M. Sloman, "The Ponder Policy Specification Language," *Proceedings of Workshop on Policies for Distributed Systems and Networks*, Springer LNCS, 2001.

[9] F. Dignum, "Abstract Norms and Electronic Institutions," *Proceedings of Int'l Workshop on Regulated Agent-Based Systems: Theories and Applications*, 2002.

[10] P. Dourish, "Accounting for System Behavior: Representation, Reflection, and Resourceful Action," Computers and Design in Context, 1995.

[11] S. Dustdar, "Caramba: A process-Aware Collaboration System Supporting Ad hoc and Collaborative processes in Virtual Teams," *Distributed and Parallel Databases*, D. Georgakopoulos (ed.), Vol. 15, No. 1, Jan.2004.

[12] FileNet: Business Process Manager, http://www.filenet.com/, 2006.

[13] D. Georgakopoulos, "Teamware: An Evaluation of Key technologies and Open Problems," *Distributed and Parallel Databases*, D. Georgakopoulos (ed.), Vol. 15, No. 1, Jan. 2004.

[14] D. Georgakopoulos, H. Schuster, D. Baker, and A. Cichocki, "Managing Escalation of Collaboration Processes in Crisis Mitigation Situations," *Proceedings 16th International Conference on Data Engineering (ICDE'00)*, San Diego, 2000.

[15] Groove: Groove Virtual Office, http://www.groove.net, 2006.

[16] C. Gutwin, S. Greenberg, and M. Roseman, "Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation, People and Computers," *Proceedings of HCI'96*, 1996.

[17] P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke, "A Comprehensive Approach to Flexibility in Workflow Management Systems," Proceedings of the *Intl. Joint Conf. on Work Activities Coordination and Collaboration (WACC'99)*, 1999.

[18] D. McGuinness, D., and van Harmelen, F. (Eds.), "OWL Web Ontology Language Overview," http://www.w3.org/TR/owl-features/, 2004.

[19] B. Michelson, "Event Driven Architecture Overview: Event-Driven SOA Is Just Part of the EDA Story," Technical Report 10.1571/bda2-2-06cc, Patricia Seybold Group, 2006. Available via http://dx.doi.org/10.1571/bda2-2-06cc.

[20] M. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. Taylor, and A. Unruh, "Active Information Gathering in InfoSleuth," *Intl. Journal of Cooperative Information Systems*, Vol. 9, No. 1, 2000.

[21] A. Ricci, M. Viroli and A. Omicini, "Role-based Access Control in MAS using Agent Coordination Contexts," *Proceedings of Workshop on Agent Organizations: Theory and Practice*, AAAI Press, 2004.

[22] S. Rinderle, M. Reichert, P. Dadam, "Flexible Support of Team Processes by Adaptive Workflow Systems," *Distributed and Parallel Databases*, Vol. 16, No. 1, July 2004.

[23] SAP: NetWeaver, http://www.sap.com, 2006.

[24] K. Schmidt, and L. Bannon, (Eds,) "Issues of Supporting Organizational Context in CSCW Systems," Lancaster University, 1993.

[25] Telcordia Technologies: ATLAS, http://www.argreenhouse.com/ATLAS/, 2003.

[26] A. Uszok, et al, "KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction and Enforcement," Proceedings of *IEEE International Workshop on Policies for Distributed Systems and Networks*, IEEE Press, June 2003.

[27] Vignette Solutions: Content Management and Portal, http://www.vignette.com, 2006.

[28] M. Weske, "Flexible Modeling and Execution of Workflow Activities," *Proceedings 31st Hawaii Int. Conf. on System Sciences*, Software Technology Track, 1998.

[29] Workflow Management Coalition, www.wfmc.org, 2006.