Event-Based Applications and Enabling Technologies

Annika Hinze
University of Waikato, New Zealand
Humboldt University Berlin, Germany
hinze@cs.waikato.ac.nz

Kai Sachs, Alejandro Buchmann
Databases and Distributed Systems Group
University of Darmstadt, Germany
lastname@dvs.tu-darmstadt.de

ABSTRACT

Event processing has become the paradigm of choice in many monitoring and reactive applications. However, the understanding of events, their composition and level of abstraction, the style of processing and the quality of service requirements vary drastically across application domains. We introduce the basic notions of event processing to create a common understanding, present the enabling technologies that are used for the implementation of event-based systems, survey a wide range of applications identifying their main features, and discuss open research issues.

1. INTRODUCTION

Event processing has become the paradigm of choice in many monitoring and reactive applications. These systems, also known as *sense-respond* systems have been developed in many different domains. Therefore, the understanding of what are events, how they are propagated, filtered, aggregated and composed into more complex events, and how events of higher levels of abstraction are derived from basic events varies widely across application domains and the different communities working on event-based systems. These communities have developed partially overlapping concepts and execution models, description languages and algebras, and have different quality of service requirements [35, 36].

In this survey we pursue four goals:

- To define the basic terms and create a common understanding across domains
- To identify the contributing technologies
- To describe a broad range of applications and their main features
- To identify problems and research issues that are common across application domains

We have chosen a set of concepts that we consider necessary to create a common understanding (Section 2). We hope that our definitions are general enough to be applicable in many application domains yet precise enough to avoid misunderstandings. Event-based systems draw on many technologies whose contributions are briefly discussed (Section 3). In choosing the applications areas (Section 4) we aimed to span a broad range, where the examples are only meant as placeholders for many similar applications. Some have a very broad scope while others are small and focused. This fact is also typical of the huge variety of sense-respond systems. Several of the applications were discussed during the Dagstuhl Workshop on Event Processing [8]. The biggest challenge is the synthesis we attempt at the end: to identify similarities across applications and domains (Sections 5 and 6) and to identify areas in which we believe research is needed to consolidate the area of event-based systems (Section 7).

DEBS'09, July 6 – 9, Nashville, TN, USA.

2. FOUNDATIONS

An *event* was defined by Chandy [9] as a significant change in the state of the universe. Since time is an inherent dimension of the universe, two observations of the universe at different time constitute two distinct events, even if no other properties have changed. Chandy's definition, however, refers to significant changes in the state of the universe, thereby limiting the infinite number of events to those that are relevant to an application. For an application, it may be relevant that an object changed its position by a few meters (*change event*), or to learn about the new reading of a temperature sensor (*status event*). Even the observation that two readings at different times yielded the *same* temperature constitutes an event. By considering time an integral part of the state of the universe, both change and status events can be modeled in a uniform manner.

Events must be observed to be reported and processed. An observation captures a discrete instance of a (possibly continuous) signal. An observation of an event carries a timestamp and descriptive parameters and is typically represented as a tuple of values. Depending on the type of event and application system, the timestamp may be just one point (point semantics of time) or an interval (interval semantics of time). Parameters may be absolute values or deltas relative to older reference values.

Events, or more precisely, their representation, must be reported to event consumers. It is generally accepted that event notifications are routed from event producers to event consumers by a notification service. The *notification service* decouples producers and consumers, and provides the routing from source to sink [36]. In the simplest form, this

may be a low-level channel into which event notifications are placed and from where they are retrieved. In this case, the envelope of the notification is minimal and streams of tuples are delivered over a fixed channel. However, the notification service may be a more sophisticated network of brokers routing the notifications based on type or content. Notifications consist of one or more event representations packaged in an envelope. Routing may occur on the content of the envelope data or the content of the notification.

Events may be simple events or compositions of simple and/or other composite events. Simple events may be individual sensor observations, method invocations or absolute temporal events. Composite events are aggregations of events. Composite events are produced from event representations and the operators of an event algebra. Two commonly used approaches to event composition exist: (1) event trees consisting of events at the leaves and the operators of an event algebra in the inner nodes [6] pioneered by active database systems, and (2) continuous or streaming queries based on the operators of relational algebra applied to subsets of streams of tuples (sliding windows) [7].

Derived events are caused by other events and often are at a different level of abstraction. For example, five failed logins with the wrong password may cause an intrusion attempt event to be signaled. Derived events involve semantic knowledge. They may be detected automatically, e.g., from a combination of sensor readings as the action part of an *event-condition-action* (ECA) rule or be raised explicitly, e.g., based on direct observation of an event by a user. Derived events are often enriched with data from external sources.

An *event-based system* is a software system in which observed events cause reactions in the system. Event-based systems consist of three essential parts: a monitoring component, a transmission mechanism, and a reactive component. The monitoring component is responsible for event observation, representation, and composition as described above.

The transmission mechanism is responsible for event notification. It is generally accepted that event notification is push-based. In push-based systems, producers disseminate information to consumers; in pull-based systems the consumer must request the information. Some authors go as far as requiring a complete decoupling of event producers and event consumers through a publish/subscribe notification service [36]. For generality, we also accept point-to-point notification of events. This implies a tighter coupling between producers and consumers, since the producers must be aware of the consumers to notify them without the help of a broker.

The reactive component of an event-based system expresses the application logic in form of rules (or other code) triggered by the corresponding events. Rules may have different formats that result in different execution models. Procedural ECA rules are fired whenever the corresponding event (simple, composite or derived) is raised. The condition acts as a guard that can be used to express more complex application logic. Only if the condition is met, the action is executed. Missing conditions are considered to be true and result in event-action rules. Much debate has occurred in the past as to the best division of functionality between events and conditions. More powerful event expressions decrease the need for explicit conditions but require more powerful event algebras. This also makes the event detection mechanism heavier and more difficult for users to use properly. On the other hand, a lightweight event mechanism can be more responsive and is less error prone but requires an explicit condition to express more powerful application logic. The decision on the trade-off between expressiveness of the event language and the lightweight nature of the event system is domain-dependent.

While the logical distinction is clear, specific implementations of event-based systems may partition the functionality differently. In particular the event composition may be implemented at the monitoring component, in the notification service, or as part of the reactive component. The decision of where to realize event composition depends on many application and environment specific factors, such as capabilities of the sensing devices, bandwidth of the communication channels, complexity of the composite events, and source of the events that are to be composed.

3. CONTRIBUTING TECHNOLOGIES

Many different technologies have contributed to the field of event-based systems. We will briefly review the contributions of these technologies.

3.1 Active Databases

Active databases were developed in the mid to late 1980s [38, 49]. Two distinct strands can be identified: relational and object-oriented active databases.

Relational active databases were limited mainly to basic database events, such as update, insert and delete. They could express conditions on either the old or new state of a relation, and the action was always some SQL statement. Today's triggers in SQL are the relational incarnation of simple ECA rules. Relational active databases introduced the notions of before, after or instead execution, meaning that the rule should be executed accordingly before, after or instead of the triggering statement.

Object-oriented active databases had a richer event type system. It included any method invocation, state changes effected through generic accessor functions, temporal events, control flow events, arbitrary user defined events, and composition of events through event algebras of varying expressiveness. The first generation of active ooDBMSs assumed a central clock, point semantics for the events, and a complete ordering of events. Active ooDBMSs introduced coupling modes to define when a rule should be executed (immediately or deferred) and whether it should be executed within the scope of the triggering transaction or as a separate transaction. If rules execute in separate transactions this can occur independently or causally dependent, in which case the triggered transaction may only begin or end execution depending on the fate of the triggering transaction. The various causal dependence modes take care of violations of ACID properties that occur when uncommitted data is made visible to independent transactions. Another major contribution was the notion of event consumption, referring to the way in which events are consumed during event composition. Four consumption modes (originally termed contexts) were defined: chronicle, recent, continuous, and cumulative. These determine that the events be consumed either in chronological order (typical in workflow-like applications), always using the most recent occurrence (typical in control applications), in form of windows (typical in finan-

cial applications) or accumulating the effects of incoming events until another event occurs (typical in inventory control situations).

3.2 Materialized Views

Materialized views can be seen as a particular application of active database principles. Views are typically subsets of a database that are defined in the database schema. They are computed on the fly from the stored base tables. As an optimization, views were materialized, i.e., stored, resulting in the need for maintaining them whenever the base data changed. Propagation of base-table updates to the materialized views was accomplished using the mechanisms developed for active relational databases [24].

3.3 Continuous Queries, Stream Processing

Continuous queries [11] can be seen as an attempt to change the processing paradigm from issuing a single non-persisting query against stored, persistent data to storing the query persistently in the database and applying it to streams of incoming data. Continuous/continual queries were expressed in variants of the SQL language modified to operate on windows [34]. These can be defined either through temporal events or through a count of incoming events. While early work on continuous queries assumed that the continuous queries would operate on the stored data and would be executed by the traditional query engine, work on streaming queries has changed the processing paradigm: queries defined in a SQL dialect, such as StreamSQL, process streams of data or events before they are placed in the database and results of this processing step are only selectively stored in the database. Many of the extensions to the relational operators and how to process for example joins on windows in continuous queries, have carried over to current products for stream processing and have been extended there for high volume applications [5].

3.4 Reactive Middleware

Reactive middleware can be traced back to the CORBA platform and the event service defined therein [25]. Modern versions of basic reactive capability can be found in the form of the J2EE message driven beans, which consume event notifications and allow the asynchronous processing of messages in the J2EE platform [45].

Reactive middleware benefited to some extent from work on active databases and the attempts to unbundle active functionality from active databases. Major insights gained while unbundling were the need for interval semantics instead of point semantics for many distributed environments, the impossibility of using a central clock and the fact that notification delays cause uncertainty. This resulted in the 2g–precedence model used in networks with bounded delay and the imprecision interval model that distinguishes between the stable past, the unstable past and present for networks without an upper bound on delay.

3.5 Message Oriented Middleware

Message Oriented Middleware (MOM) covers basically notification middleware built on the principles of the publish/subscribe paradigm. Publish/subscribe systems come in many different flavours, both centralized and distributed. A common distinction is based on the information carried by the notification, and whether the content of the notification is used for routing or only the information on the envelope of a message [15].

Channel-based publish/subscribe is the most simple. Producers place their notifications into a channel and consumers subscribe to a channel of interest.

Subject-based publish/subscribe, pioneered in the 1990's by TIBCO [48], defines subject hierarchies according to which messages are classified. A combination of subject hierarchy levels with the use of wild cards allows for reasonably powerful subscriptions. One disadvantage, though, is the relative inflexibility of subject hierarchies.

Content-based publish/subscribe uses the content of a message to route the message from producer to subscriber [43]. Filters are placed as close as possible to the source to minimize traffic. Predicates of different degree of expressiveness can be specified. However, the more powerful the predicate language and the more fine grained the filters are, the more critical it becomes to control the size of the routing tables. This in turn requires merging of filters.

Concept-based publish/subscribe finally addresses the problem of heterogeneity [12]. All the previous approaches to publish/subscribe assume a common understanding of the name space used. If this is not the case, then an additional layer of mediation that resolves semantic conflicts based on an ontology service can be used. Concept-based publish/subscribe uses predefined contexts. If notifications are to be routed within a common context, i.e., publisher and subscriber use the same context, no additional mediation is needed. If publisher and subscriber use different contexts, an additional mediation step is needed. Concept-based publish/subscribe can be implemented on top of any of the other publish/subscribe methods.

JMS as a de facto standard for MOM provides both queuing and publish/subscribe type interaction. JMS's flavour of publish/subscribe, *topic-based* publish/subscribe, is a variant of channel based publish/subscribe with additional predicates definable on the envelope data.

3.6 Identity and Location Detection

Common tagging systems use barcodes and *radio frequency identification* (RFID) tags [21]. Barcodes can be simple linear codes or 2D. Linear barcodes typically encode a numeric code that is optically scanned and the information attached to the barcode is looked up. 2D barcodes can encode larger information. RFID tags consist of a circuit and an antenna. Passive RFIDs are activated by the energy of the reader and with the induced current they emit a signal via radio waves containing the tag identifier. Associated information must be looked up. Active RFID tags can have their own power supply and may store additional data. Common to all tagging systems is that a reader produces a tag detection event that is interpreted either at the device controller or in the backend system. Typically, position information of the reader is associated with a detection event.

Location detection in mobile systems typically relies on the widely available Global Positioning System (GPS) in a mobile device, or stationary beacons that communicate with simpler mobile devices. GPS observes a user's outdoor location continually following a given frequency. Other observation semantics can be built on top of this continuous observation. GPS delivers latitude and longitude of the current device location. Beacons support passive client devices. A beacon is installed in places of interest; devices in close

proximity receive its signal and can thus identify their logical position in relation to the beacon

3.7 Wireless Sensor Networks (WSN)

WSN consist of large numbers of small computing devices with one or more sensors on board that communicate over radio frequencies forming ad-hoc networks [1]. Sensed data can be filtered and aggregated in the network as it flows from the source to the sink. Energy consumption is a major issue, requiring energy conscious routing and minimization of transmitted. Because of the unstable nature of the communication and the risk of failure of individual nodes, alternative routing strategies and redundancy are typically provided in WSN. In modern WSN, nodes may be stationary or mobile, and heterogeneous sensor nodes are becoming common. The boundary between basic sensor nodes and active RFID tags is blurring.

MANets are mobile ad-hoc networks that form among mobile devices (PDAs, smartphones, laptops) using wireless technologies (e.g., WLAN). The networks have a very dynamic topology because their nodes (the mobile devices) may be frequently changing location, causing connections to fail and form newly. Moreover, the network nodes are unreliable as the mobile devices may be turned off or suffer from energy constraints.

Sensor Fusion has its origins in military command and control systems in which data coming from multiple sensors must be combined to extract the information. This information is often related to tracking of vehicles, vessels or missiles, and for friend-foe-neutral identification. Sensor fusion often combines automatic processing of sensor data with humans in the loop.

3.8 Data and Web Mining

Data Mining is the process of finding correlations or patterns among attributes in large datasets using techniques rooted in statistical analysis or artificial intelligence [50]. Typical data mining techniques include: neural networks (for trend analysis based on past performance); association discovery (to identify similar occurrences within data sets and express those as rules with a confidence factor, e.g. associations based on medical data); classification (grouping data with similar characteristic defined by analyst, used frequently in customer retention); clustering (grouping of data sets based on discovered similarities, used frequently in opinion analysis); and sequential discovery (discovers repeat occurrences over predefined time window, e.g. buying patterns, useful in fraud detection). Data mining is increasingly applied to streaming data [18].

Web Mining uses the web as data source. Techniques are based on clustering (natural groupings of users and pages), associations (URLs being accessed together), and sequential analysis (the order in which URLs are accessed). Clickstream analysis is a form of data mining of the URLs accessed by a user while at a given web site. It can be used off-line for analysis of customer behavior and for system performance analysis and tuning, or it can be used on-the-fly as part of recommender systems.

4. APPLICATIONS

We analyzed 17 applications for their event features. The applications are discussed here and their identified features in Section 5.

4.1 Content-awareness: Mobile Tourist Information System

Mobile tourist information systems are location-based systems. Typically they provide their mobile users with feedback about their current location (e.g., indicated on a map) and additional information about tourist sights that are related to the user's location. The information may use travel information and feedback previously given by this or similar users. The systems are an extended form of mobile navigation systems. The user's location or change of location are triggering events for (1) display of the triggering information (e.g., current user location on map), (2) retrieval and display of information relevant to the current location, and (3) prediction of the user's future location (e.g., for pre-fetching and caching strategies). The location event may trigger an action on each event, after a time-span, or after sufficient change of location. Potentially every event may trigger an action and has to be analyzed for the system. However, missed events are tolerable.

A system may be implemented as a context-aware system considering additional information such as the current time, the user's interest and background, and their travel history [29]. The travel history is in fact an event history – the system thus reacts differently depending on a combination of past events. For example, the system may decide not to display certain information because it has been displayed before (assuming the tourist is familiar with the place) or it may recommend certain sights based on the current location and the past locations of similar users. Changes in context are reported as events.

Mobile tourist systems are implemented as distributed client-server systems, as peer-to-peers systems, or as standalone client software. The software architecture influences the available features and potential issues [27]. The users are mobile; typically the location sensors are mobile with the users (e.g., GPS). Other options are mobile receivers reacting to stationary beacons, or mobile readers sensing stationary or mobile RFID tags. The locations of users and items may be identified by their location (geographic or layout) or in relation to another object (e.g., to the car); physical or logical coordinates may be used. The events considered in the system are status events, such as the current position reading of the GPS, or a combination of position and time (and other contextual information). The time may be measured using a global time or a relative local time. Different sensors may use different time systems. A combination of software services may have to collaborate on the user's mobile device; context acquisition may use a variety of forms. Systems have to incorporate changing sensors and a changing mobile environment.

4.2 Baggage Management

The main goal of baggage management in an airport is to manage the transport of the luggage via conveyors, carts, and planes to the right destination. To increase the degree of automation, luggage is tagged at the Check-In, traditionally with barcodes, nowadays more and more with RFID tags [14]. When baggage passes a tag reader device, an event containing the position and time is reported. Based on these events, the luggage routing is managed by an event-triggered environment [37]. Event information has to be made persistent to allow later tracking of lost baggage.

A central enterprise system per airport is storing all the

tracking events and combines them with the flight data of different airlines. Based on this information, routing plans for the baggage are specified and updated, e.g., the baggage has to be stored until the state of the airplane is 'ready for loading' (where the state change of the plane to 'ready for loading' is another event). Additionally, several other types of data have to be taken into account, for example, security checks, passengers on board, and containers where the luggage is stored. All these data has to be accessible in real-time. For example, the baggage is only allowed to be loaded into the airplane, if it has passed all security checks successfully.

Visibility and security of baggage events are challenging issues to be addressed. For example, the airline A is not allowed to see the baggage events of passengers of airline B and vice versa. But the security staff is allowed to access all the data, but not to add new passengers or modify flight data. On the other hand, the airport service staff has access to all tracking data and is allowed to trigger events such as airplane 'ready for loading'. Visibility controlling concepts, such as scopes, are needed [17]. Higher level events, for example, a plane arriving at the gate, may trigger a whole chain of activities.

4.3 Traffic Monitoring

Traffic monitoring is a rather broad application that encompasses a variety of tasks [30]. These range from simple counting of vehicles during specific time windows for the purpose of traffic planning and contingency road management to the identification of individual vehicles for levying of tolls and control of traffic restricted areas all the way to searching for particular vehicles in support of police work. Depending on the scope of a traffic monitoring system, the complexity of the events and their associated information varies greatly.

Traffic monitoring systems may be based on passive tags detected by active readers, they may involve active devices emitting beacons or some positioning signal or they may be based solely on active detectors, such as cameras combined with image recognition software.

The scope of a traffic monitoring system determines the kind of event detection mechanism used. Conversely, depending on the available event detection mechanism a traffic monitoring system may be limited to fulfilling a single task, such as access control to toll roads or bridges with monthly billing (typical of passive tags) or may allow multiple functions, such as anonymous counting of vehicles in a time window, the control of access to inner cities or the surveillance and tracking of specific vehicles (typical of camera-based systems).

The privacy and security issues related to the distribution of events may vary. Simple toll control systems requiring the installation of a tag or beacon on the part of the user implicitly have the user's consent to be monitored, and therefore minimize any privacy concerns. Camera-based multipurpose traffic monitoring systems require a strict differentiation of who is permitted to receive certain events, whether they must be anonymized or not, and for how long certain events may be tracked and stored. This has profound implications on the kind of event aggregation permitted (tracking) or required (anonymization), and on the security requirements posed on the event distribution mechanism, e.g., a secure publish/subscribe system.

4.4 Environmental Monitoring: Avalanche Warning System

Avalanche warning systems combine information about snow conditions and weather information to predict avalanches and send out alerts and warnings [41]. For local avalanche warning systems, it is necessary to supplement meteorological and manual observations by snow parameters measured automatically in the release zones and at representative locations. The systems have to supply reliable data on the development of dangerous snow covers between and during storms. Typical approaches use a variety of sensors: microwave snow radars, ultrasonic snow depth gauges, snow temperature and IR-surface-temperature measurements, combined with measurement of reflected short wave radiation, air humidity and temperature measurements. Alternatively, an area can be monitored by simply distributing sensor nodes to cooperatively determine the state of a field. Each individual sensor node is equipped with a variety of application-specific sensors to sample its physical surrounding, a microcontroller to allow for further processing, some secondary storage and a transceiver to enable ad-hoc wireless communication with other nodes in the network.

A large number of such mobile devices is distributed on the mountain, where they form an ad-hoc network. With their processing capabilities at the sources of data generation, these embedded devices can be used as a filter on the incoming data streams of their sensors, for providing event recognition locally on each node, or for complex, distributed event processing using spatio-temporal event pattern matching. Computation is therefore performed either distributed at the sensor level or centrally [47]. Events are status reports. Only a few event constellations are sought out in the filter process to trigger alert notifications. Dynamically changing weather conditions to be monitored include amount, thickness and weight of snow, temperature and pressure, but also the history of these conditions. Triggering events are combinations of concurrent and historical event reports – avalanches are influenced by snow consistency and temperature as well as the progression of the weather and snow situation over a time period.

4.5 Information Dissemination

Automatic systems for information dissemination have been used by conventional libraries for about 50 years; they were called awareness systems (SDI – selective dissemination of information) and served for synchronization of library catalogues. Metadata of new publications is filtered and distributed to library catalogues. Digital libraries use alerting systems that follow similar principles, with extended functionality [2]. A digital library consists of collections of documents that may be distributed over several heterogeneous library installations. A Digital Library (DL) may use a database for storing metadata of documents but typically not for the items themselves. An alerting system may inform about changes in a set of documents, such as new documents, new parts of documents, new collections as well as changed documents or metadata or deleted entries or documents. The system may inform of changes or support direct ingest of information via an awareness service. Digital library documents are not just textual information but also multimedia items; examples are images, such as maps or photographs, structured information such as sheet music, and music recordings in various storage formats.

The observation of events is crucial – it is performed either by the digital library (integrated service) or by a third party that may combine information from various sources (meta-service). Event observation faces a number of issues as in-place document updates may not be observable by an external service and updates may be implemented as delete followed by insert operation, which leads to false event messages. Event detection requires rich concepts of works and documents to be supported by the digital library system and the alerting system. It needs to identify the work (Shakespeare's plays or Mozart's piece) in addition to the document or item and be able to identify qualitative metadata such as copies, editions, recordings, productions. Combining events from a variety of sources beyond simple event occurrences requires in-depth knowledge of the DL systems. Where a collection is spread over different servers, or different collections are brought together as a virtual entity, we not only have to inform users when the content on one machine changes, but also the other machines as well. This is made particularly difficult as often DL do not cooperate and the connecting networks are therefore unstable and fragmented. In a DL environment it is critical for the users to have a single homogenous access point to all their profiles and alert data.

4.6 Blog Information Mining & Dissemination

The ever-increasing flood of news and posts to regular web-sites, and in the recent past to blogs, makes alerting of new items of interest and presentation of this information in the form of dashboards a necessity. The existence of tracking services, such as spinn3r[1], that crawl for new blog postings and offer APIs to get at regular intervals the new postings, can be exploited as event detectors. Other services, such as Calais[2], provide text analysis and quite sophisticated annotations. Calais not only provides an analysis of the terms found in a submitted text, in this case the text of a posting, but also enriches the relevant terms with additional information from available sources. For example, a posting about the merger of two companies will be enriched with background information about the two. The analyzed text and the annotations can then be further processed to extract other high level events and report the findings in an aggregated manner, for example, as a personal or thematic dashboard to the subscriber. Filtering and annotation of news paper articles for health-care professionals relies on similar concepts of event enrichment [28]. The processing of the streams can be done using existing event processing engines, such as the Avaya EP engine[3]. Base techniques for simple forms of update monitoring are RSS or atom feeds, ajax and various readers implemented for monitoring and displaying feeds. Blog mining is a simple version of mining of heterogeneous information mining [10] and web-based information dissemination methods discussed in [42].

4.7 Fraud Detection

Fraud detection is a highly profitable application of event processing and has been studied and refined for some time [16]. A typical class of fraud is known as superimposition fraud, in which a fraudulent user gains access to a legitimate user's account or service and uses it alongside the legitimate user. Examples of superimposition fraud are credit card fraud, calling card fraud, and cloning fraud in mobile telephony networks. Fraudulent use is detectable whenever the legitimate user has a fairly regular usage pattern. A cell phone being used for business purposes with a call pattern of one to five minutes during business hours that suddenly shows long calls at night shows such an anomalous pattern. Other techniques used to detect fraudulent use require the temporal and spatial correlation of calls traceable to the same phone. Calls made within too small a time window from two distant locations strongly indicate cloning fraud. Superimposition fraud can be detected after the fact by mining persistent call data or it can be detected by on-the-fly processing of events. Mining persisted events does not prevent the fraud but allows the company to avoid false charges to the legitimate users and aggravating them as well as saving call center processing costs.

Superimposition fraud detection is typically done on large volumes of fairly simple and homogeneous event records, conditions that are typically well suited for stream processing. Other kind of fraud, for example, detection of insider trading, requires the correlation of unusual trading patterns from ticker data with external information, such as earnings reports, merger and acquisition or new product announcements. This enrichment information may in turn be events from news feeds or other information dissemination systems. Tightly coupled with the topic of fraud detection is the problem of compliance management. Compliance management deals with the pre-trade validation of transaction conditions and could be considered fraud prevention. Pre-trade conditions must be continuously monitored and compared for compliance with internal and external policies that may be derived from regulations, such as those established by the SEC, or legislation, such as Sarbanes-Oxley or the US Patriot Act.

4.8 Financial Applications

Many applications in the financial domain are event driven. Examples are algorithmic trading, transaction cost analysis, real-time profit and loss analysis, and compliance management, which has been touched upon under fraud detection. Algorithmic trading has become highly dependent on the exploitation of timely information about worldwide trades. Ticker services provide high-volume streams of homogeneous transaction records from various markets that must be filtered. The volume of ticker feeds that must be processed is in the hundreds of thousands of events per second, thus making the scalability of the event processing engine a primary concern. Algorithmic trading attempts to obtain favorable positions based on the real-time information from the various markets filtered out from the event stream and subject to trading policies that are based on value added analytics, such as Volume-Weighted Average Price, the ratio of the value traded to total volume traded over a particular time horizon. The computation of these value added metrics requires the definition of time windows and the tracking of transactions over this window to compute the desired metrics. Algorithmic trading also requires making the event streams persistent and the capability to replay event streams. This is often done for testing of new trading strategies on old trading data to compare performance before going life with a new strategy.

[1]See `http://spinn3r.com`

[2]See `http://opencalais.com`

[3]See `http://www.avaya.com/gcm/master-usa/en-us/products/offers/event_processor.htm`

Real-time Profit and Loss (P&L) is emerging as a new application of stream processing systems [20]. The volatility of the markets and sudden spikes in activity may lead to unexpected exposures of companies and may confuse the algorithmic trading systems. Daily P&L is inadequate under these circumstances. To compute real-time P&L it is necessary to integrate life feeds from multiple markets and across assets, consolidated feeds, such as those provided by Reuters or Bloomberg, direct feeds from stock exchanges, internal feeds, and tie into the work-stream for processing trades. Different streams must be homogenized, cleansed; sometimes currencies must be adjusted, and common reporting bases must be created. Based on the combination of trades and market conditions, the individual valuations are computed and aggregated to compute profit and loss. Individual positions may be rolled up to different levels of aggregation (book or portfolio level) and the corresponding alerts may be generated to traders.

4.9 Supply Chain Management

Supply chain management (SCM) integrates the processes involved in the timely provisioning of goods. This involves reacting to low stock of parts or supplies, ordering, integrating the logistics process and tracking the status of shipments, warehouse management, etc. Many modern SCM systems depend on RFID technology. Because of the cost of RFID tags this technology is mostly applied when dealing with high value products or on aggregates, such as pallets of lesser value products. Events are typically RFID tag readings that must be supplemented by information from product or shipment databases. Events are captured by readers and processed along the chain between the peripheral detectors and the backend systems [23].

Typical processing includes filtering and elimination of duplicates, aggregation, enrichment through database lookups, correlation of tag readings and reader positions, maintenance of traces, all the way to triggering of business rules. Major problems are derived from noise in the detection process, resulting in the use of multiple readers to reduce false negatives at the expense of added cost for the removal of duplicates, uncertainty in the aggregation process where it is often difficult to distinguish between a false negative and a missing object, and the need for processing business rules as close as possible to the periphery to allow for fast response and to improve scalability. Alternative sources of events in SCM systems are database events that may, for example, trigger the reordering process when quantities on hand fall below a threshold. Modern SCM systems are expanding beyond simple RFID technology and include other sensors, for example temperature sensors to insure that the cooling chain is not broken when dealing with perishable goods or for detecting tampering with containers.

4.10 Railway Scenario – Cargo

Freight-cars will be equipped with a variety of sensors that may reach from temperature sensors to detect overheating of axle bearings that might lead later to derailments, to sensors signaling the position of a given freight car both in absolute terms as well as relative to its neighbors in a given train, and sensors monitoring the condition of the cargo [40].

Event signals from a large number of sensors need to be detected, aggregated and distributed via a heterogeneous wireless sensor network. One of the requirements on the overlay network distributing the events from producers to consumers is the need to form groups of nodes. Groups of sensors must be definable in an easy manner, preferably declaratively, based on static and dynamic predicates. A typical grouping consists of all the nodes belonging to a single train. Trains must communicate with the control stations as a unit but communication must exclude neighboring trains that may be standing on adjacent tracks in a station. For security applications, however, events must be propagated across train boundaries to all reachable nodes. Controlled visibility of events is a primary requirement in this application.

Because of the safety-critical nature of some events and the privacy requirements of others it is important to offer secure event notification. Therefore, a secure publish/subscribe mechanism is required. The railway scenario has many features in common with sensors and event-based processing in containers.

4.11 Piping Information to the Cockpit

While today's communication with the airplane cockpit is mostly based on voice communication between tower and cockpit or notes-to-airmen (Notams) on paper handed to the cockpit personnel, the future communication will be through electronic distribution of information to the cockpit [22]. Two basic requirements for the distribution of relevant events to the cockpit are (a) that no relevant information may be lost, and (b) no superfluous or irrelevant information should be delivered to the cockpit. Relevance is defined with respect to a flight and the proposed flight route and the in-flight changes due to weather, air traffic control instructions or other external factors.

Relevant events that must be communicated could be changes to a runway at the destination airport, relevant weather conditions, and other aircraft on collision course. Some of these events are short-lived but have high priority while other events, such as runway closings, may be planned in advance, have a begin time and a validity interval. All instances of flights to that destination airport must be notified of the event during the duration of the validity interval.

Besides the delivery guarantees required in this application, the event notification mechanism must fulfill two specific requirements: Event routing and delivery must occur according to geo-spatial filters defined for the relevant information, and event notifications must be transmitted to future subscribers, i.e., flights that are instantiated after an event has occurred. This property requires the persistence of events and symmetry between subscriptions and the advertisements that announce the types of available event information. A combination of push- and pull-based interaction may be appropriate here.

4.12 Health Care: Health Monitor

Health care has many applications for event-based systems; a personal health monitor is one example application [26]. A health monitor is a personalized system that allows a person and their care givers to monitor the person's health status. Health monitors may be particularly useful for chronically ill people as well as for elderly citizens [31].

The data may be captured from mobile sensors at the person as well as from stationary sensors in their home. Alarms are set up to alert the person and, if necessary, a remote care-giver. Sensors automatically capture health-

related data such as heart rate and blood pressure, the location of the person in reference to a room, or the intake of medication. In addition, specific regular measurements for particular health conditions are performed by the person and the results are filtered for emergency situations as well as kept for long term observation. The data needs to be protected from tampering, from external unqualified access, as well as being kept safe for long term storage.

Health monitors offer many challenges derived from the high volume of low level events and the need to derive higher level events that must be propagated. This application is particularly sensitive to false negatives and false positives. False negatives may lead to loss of life in the extreme and false positives may trigger unneeded and costly emergency responses. These liability issues are a major hurdle for the widespread adoption of health monitors.

Security is a major concern, especially as monitoring data may give rise to increased health insurance premiums or attempts to limit services. Although publicly perceived as a major threat, privacy is not reported to be a major concern for senior citizens and the chronically ill who perceive that the benefits outweigh the risks and the fear of being helpless after suffering an incapacitating health event that is not noticed.

4.13 Ambient Assisted Living, Smart Homes

Ambient assisted living aims at prolonging autonomous living of a rapidly aging population through technological support. It subsumes many of the functions already described above as part of health monitoring but includes many more, such as alarms when a person tries to leave home without turning off the stove, voice activated devices or special control panels aimed at persons with fading eye sight, and emergency buttons. Smart homes on the other hand do not target a particular age group but have been proposed often for the media and technology savvy to provide remote access to smart appliances, proactive appliances, seamless personalized lighting and media coverage, network access and application migration as a person moves around the house, automatic synchronization among stationary and mobile devices, etc. Smart homes also provide a large amount of sensors that support energy efficiency by controlling shades based on time of day and season, sophisticated cooling and heating patterns, or security. Multiple projects addressing different aspects exist in this domain, for example described in [13, 46].

Events in smart homes are often based on detection of presence of persons and/or devices in a room, external (weather) conditions, or temporal events exploiting known behavior patterns of the occupants. Event composition across heterogeneous sensor readings is needed. Simplifying factors are the relatively low volume of events and a fairly self-contained environment. A major problem is the integration of new devices (i.e. consumers and producers of events) into the environment due to standardization problems and the widely diverging life cycles of smart devices and the physical plant (i.e. house and integrated sensors).

The commercial application of smart home technology is found in facility management. In the same way, events are related to locations (e.g., temperature somewhere, motion detection) and buildings are equipped with (stationary) sensors. Commercial buildings are often used for varying scenarios, requiring changes in event handling and setting. In addition, large numbers of sensors of different types need to be supported. The applications typically require reliable logging and persistency for insurance purposes.

4.14 Smart Cities and Infrastructure for Ambient Intelligence

The vision of *Ambient Intelligence* (AmI) calls for the vanishing of the computational infrastructure and instead the establishing of pervasively available, context-aware services [3]. This in turn requires from the infrastructure to provide seamless event processing capabilities across spaces and domains, i.e., event detection and composition, reactive capabilities, support for indoor and outdoor positioning, context and user-profile management and matching, support for privacy and security, and support for mobility and event delivery with controllable quality of service. Because of the heterogeneous environment and the different life cycles of the technologies involved, support for heterogeneity and multiple technology stacks is a must. Event processing is also at the heart of the Self-managing properties these systems must exhibit, since any solution depending on manual and/or centralized management, configuration and adaptation will fail.

Rich context definition is necessary since context must include, besides location, semantic annotations, user preferences, time, resource availability and even social context. Many of the challenges for event processing in this scenario also stem from the scale of such a system, for example, if the infrastructure is to cover a whole city, including public spaces and means of transportation. The necessity to provide context-aware services independently of location and in a seamless manner may provide us with the killer application for event-based processing.

4.15 Autonomic Computing, Self-X Systems

Autonomic computing refers to the ability of systems to self-configure, self-heal, self-optimize and self-protect. These so-called *self-CHOP* properties are at the core of any self-managing system. Self-configuration refers to dynamically adapting to a changing environment according to defined policies. Changes could be the addition or removal of components or adaptation to changes in workload. Self-configuration aims at providing stable system characteristics. *Self-healing* refers to the ability to discover, diagnose and react to disruptions. Corrective actions could be policy based and involve a component changing its own state or effecting changes in other components of the environment. Self-healing aims at making the system as a whole more resilient. *Self-optimization* implies the monitoring of resources and tuning of the processes and components to meet the users' and system's demands. *Self-protection* refers to the anticipation, detection, identification and reaction to threats. The CHOP-properties are not orthogonal and several interdependencies exist. Corrective actions performed as a self-protecting measure may be common to self-healing and the result of self-optimization may consist in system reconfiguration.

Self-managing systems are typical sense-react systems, in which the state of the system is continuously monitored and compared to acceptable system states. Whenever a threshold is crossed for any of the monitored parameters or a certain pattern is detected, the system invokes predefined reactions attempting to return to a normal operating condition. From an event-processing point of view, self-managing systems require detection of simple events and their compo-

sitions, both over time (time series) to observe trends and across event detectors. Monitoring may occur on derived system parameters, such as utilization, or direct observations, such as arrival rates. Reactions may be triggered by simple events, such as the access to a honey-pot or honeycomb or it may be triggered by the observation of sudden peaks in activity. Self-CHOP properties are much better studied in closed, centralized systems but are a necessary component of large, distributed and heterogeneous infrastructures for ubiquitous computing and ambient intelligence.

4.16 Threat Detection (Dirty Bomb, Spills)

Several scenarios are being considered that involve similar requirements. Threats may be deliberate, such as is the case with a terrorist carrying a dirty bomb in a crowded public space or involuntary, as is the case with accidental spills of dangerous substances [51]. In both scenarios the common situation is that the source of the threat and the extension and intensity of the radiation or concentration of the toxic substance are to be determined through sensors mounted on autonomous land and air vehicles. These vehicles can get close to the source of the contamination and can send measurements to a base station. In addition to the sensing of toxic substances or radiation, the autonomous vehicles are equipped with their own on-board sensors for navigation purposes. Typical instrumentation for autonomous navigation include, but are not limited to, LIDAR, infrared sensors, GPS and other complementary position determination, onboard terrain models, image processing, etc. The measurement events may have to be communicated either directly to a sink, or they may have to be relayed by other nodes. Direct communication among nodes may also be needed for rapid homing in on the location(s) of the source(s). This implies on-board processing as well as centralized processing that allows further correlation of the measurements. This kind of application typically involves a human in the loop at the central control point.

4.17 Gaming

The gaming industry is rapidly becoming a major economic force that surpassed the movie industry in revenue. Large massively parallel multiplayer online games (MMOGs), such as World of Warcraft, may have several million players. Players react to game events, i.e., the actions performed by other players. The game events may be simple movements of the avatars, attacking or being attacked or solving some task and acquiring certain weapon or capability. Game events must be notified to those players who are affected by them. Notification must occur prompt and to all affected parties within tight timing bounds for fairness reasons. The typical architecture of MMOGs is a classical client/server architecture where game events are sent to the server controlling a shard of the game world and from there they are propagated to all the interested players within that shard. Event distribution is conceptually a centralized publish/subscribe mechanism with changing subscriptions as the players move around in the virtual game space [32]. Game events must be made selectively persistent: positions or position changes must only persist to restore a certain (recent) game state and old position events can be deleted. Acquired powers or weapons must persist indefinitely or until lost or traded. Game events must be protected and not tampered with. This is particularly needed to avoid cheating. In addition to avoiding the tampering with game events, analysis of event streams can be used for detecting unusual activity patterns that might be indicative of cheats. While the event aggregation that occurs as part of the game is quite simple, the aggregations needed for cheat detection may involve multiple events over relatively large windows. Some game MMOG providers are teaming up with providers of stream processing engines.

Alternative architectures, mostly experimental at this stage, consider using a Peer-to-Peer infrastructure instead of the classic client/server architecture. A P2P substrate solves the scalability problems that affect classic client/server systems but incurs new problems for maintaining game consistency and controlling cheats. Event distribution now occurs through a distributed publish/subscribe mechanism on the P2P substrate.

5. FEATURES

The goal when analyzing an application is to extract the key features a system must provide. Some features are essential while others are just nice to have. However, all too often the required features are considered only in the immediate context of an application. To provide adequate platforms or development frameworks for event based systems we must abstract from the immediate application and identify the basic primitives and abstractions that are required. From the applications we reviewed in Section 4 we abstracted the most relevant features. These are briefly described below and then correlated with the applications in Table 1.

Note that for each application domain, we necessarily had particular examples in mind, and discrepancies with each reader's perception of an application are therefore possible. An individual application may also require fewer features than a set of applications in a domain. For example, for a particular vehicle-based navigation support system it may be perfectly acceptable to rely on GPS-based absolute positioning only. A navigation aid for pedestrians must also be useful indoors and may depend not only on other positioning mechanisms for absolute position but also on relative and logical positions as well as landmarks for localization. In the broad category of navigation aids these distinctions necessarily are lost.

The features we identified as relevant are briefly described below together with examples taken from the analysed application domains.

- **Temporal Events** refers to the need for absolute and/or relative time events at varying granularities. Temporal events can also be used for the definition of windows on event streams.
- **Absolute Position Events** refers to the direct use of coordinates as position events. Absolute positions may have to be converted from logical positioning, for example, converting from (RFID) indoor positions to location events using GPS coordinates.
- **Logical Positioning** refers to the use of logical position identifiers and may require a lookup or conversion from absolute position. These are often found in applications relating to human dimensions, such as a user being in certain location in ambient assisted living situations or a suitcase being near to a tag reader (baggage tracking).

- **Spatio-temporal Correlation** refers to the correlation of spatial and temporal observations and is essential in tracking applications such as those for monitoring the location of cargo on its journey over time.
- **Other Forms of Contextual Information in Events** may be, e.g., social context, group memberships, or visual range in games.
- **Change Events** refers to the need of recognizing significant changes of the environment as distinct events. These were observed in all the analysed applications.
- **Status Events** refers to the need of recognizing that sensor readings at different times, with or without a change of sensor reading value, constitute significant events. These events are observed, for example, when tracking cargo: alerting to the fact that some items stopped moving and where left behind.
- **Interval Events** refers to events that have a certain duration and may carry a validity interval, such as a road being closed due to an accident or a runway due to scheduled repairs.
- **Event Sequencing/Out of Order Events** refers to the need to establish the sequence of events, either causal or temporal, and the need to deal successfully with events that arrive late or out of order. This is relevant in the processing of stock ticker data as well as any application with transmission delays.
- **Homogeneous Aggregation** refers to the composition of homogeneous events, typically arriving as streams of tuples from one source or multiple sources producing the same type of events. Two applications exhibiting homogeneous aggregation are stream processing of stock data and avalanche warning using a large number of sensors of the same type.
- **Heterogeneous Composition/Fusion** refers to the need of composing events from multiple, heterogeneous events sources, typically different kinds of sensors. Event fusion is required in applications that combine a variety of different aspects, such as in smart cities and ambient living.
- **Derived (higher abstraction) Events** refer to the need an application has to infer events of a higher abstraction level from the basic events that are detected, for example, the interruption of the cooling chain during the transport of perishable goods based on trends in temperature readings.
- **Event Enrichment** refers to the combination of simple or composite events with external information to derive more abstract events. External information may be contextual information and may come from external sources, such as newsfeeds. Event enrichment is typical for information mining applications, such as blog mining and information cockpits.
- **Event Re-use** refers to the possibility of an event being used in multiple compositions and the event's ability to trigger multiple reactions. This feature is required in application scenarios that integrate multiple individual applications, such as traffic or health monitoring.
- **Outlier Handling** refers to the ignoring or identifying of outliers. Some applications specifically try to detect outliers (e.g, environmental warning) while others suppress them as miss-readings (e.g., tourism).
- **Early Filtering** happens near the source with forwarding of only a fraction of the events vs. forwarding the complete stream of events and filtering at or near the sink (late filtering). Typical for early filtering are WSN-based applications while financial applications are typical for late filtering.
- **Event Purging** refers to the life-cycle of events and the need or possibility of deleting events from the system. Traffic surveillance, for example, imposes legal limitations on the life of events.
- **Event persistence** is the need of making events persistent for later analysis.
- **Audit Trail** refers to keeping an uninterrupted and persistent trail of events for post-mortem analysis. This is the case with financial and some medical applications.
- **Event Propagation/Notification** identifies if decoupled asynchronous event notification is required by an application.
- **Delivery Guarantees** are the guarantees the notification mechanism must give to an application, may range from best effort to exactly once, and may or may not be time constrained.
- **False Positives** refers to the sensibility of an application to false positives and the potential consequences of false positives as identified in baggage tracking.
- **False Negatives** refers to the sensibility of an application to false negatives and the potential consequences of false negatives. These are particularly dangerous in medical applications.
- **Transaction Processing** refers to the need of integrating event processing with transactional processing.
- **Point of Processing** refers to the processing of events and can be either distributed, in network processing, or centralized.
- **Heterogeneity of Platform** refers to the degree of heterogeneity of the platform that processes the events. While centralized systems are by nature rather homogeneous, a wireless sensor network consisting of a single kind of sensor nodes and a single more powerful sink is also quite homogeneous while a sensor network consisting of multiple types of sensors and several intermediate processing stages will be highly heterogeneous.
- **Volume** refers to the number of events that must be processed per time unit. While it is a major differentiator among applications its effect also depends on the processing platform, since a large global volume may traduce into small local volumes on highly distributed platforms.

- **Security** refers to the need of securing the transmission and processing of events by guaranteeing the authenticity of a source, the secrecy and integrity during transmission, and the secure archival of events. Health and traffic monitoring are prime examples.
- **Privacy** refers to the judicious handling of events to protect the personal sphere of individuals and to protect their right to informational self-determination.
- **Mobility of Event Source** refers to the change of position of the event source and the need for wireless communication.
- **Mobility of Event Subscriber** refers to the change of position of the event subscriber and the need to adapt event delivery based on changing position and context.

6. PLATFORMS

We analysed platforms used to implement the reviewed applications and evaluated them according to their support for the features identified in Section 5. Our analysis shows some clear distinctions along the lines of homogeneity/heterogeneity, volume of events processed, the nature of the events, the importance of the event distribution mechanism, and the need for mobility and the concomitant use of wireless communication.

6.1 Stream Processing Engines

These are typically used for processing very large volumes of homogeneous events that are provided in the form of continuous and high-volume streams. Filters and continuous queries are expressed in a SQL-based language. Scalability and non-blocking behaviour are of paramount importance. Stream processing engines exhibit a high degree of centralization and often run on large mainframes or clusters. Typical application domains are fraud detection, financial applications, and cheat detection in gaming.

6.2 (Wireless) Sensor Networks

These are typically used in small, well-contained applications in which homogeneous sensors are connected in the form of multi-hop networks to a single sink. These platforms at present do not scale and are typically used for low volumes of events. Processing is done mostly in the network with relatively simple filtering and event aggregation and some composition across heterogeneous sensors. Communication is wireless, low bandwidth, unreliable, and often the limiting factor. Typical applications are environmental monitoring, such as the avalanche warning system, and threat or contaminant detection at the high end including mobility of the nodes.

6.3 Messaging Systems

These systems are based on reliable and scalable message delivery systems that can connect stationary and/or mobile event publishers and subscribers. Event filtering, composition and routing occur in the broker network and the main goal is to decouple event sources and event consumers. Most information dissemination applications use messaging systems of different kinds as platform. Examples are information dissemination into the cockpit using a geo-spatial publish/subscribe system, conventional information dissemination systems, or blog alerts, in which the broker nodes carry out significant event enrichment.

6.4 Mixed-mode Systems

These platforms include a wide variety of nodes, ranging from simple tag readers and sensors to high end servers. Mixed-mode systems are typical for environments in which multiple smaller applications are integrated. Among the reviewed applications, the infrastructure for smart cities, large scale health monitoring and care systems, integrated traffic monitoring and management systems all require mixed mode platforms. The event streams are heterogeneous, both in nature and volume. To scale, the communication must be based on messaging systems, enrichment and event derivation may occur at different nodes in the network, and stream processing engines may be required at selected nodes to detect complex patterns on many event streams. The high volume in mixed mode systems is often the result of many converging low volume streams rather than a few high volume streams. The combination of mobility, heterogeneity, and an extreme distribution compounds the problems these platforms must deal with.

7. RESEARCH ISSUES

One of the lessons learned from the analysis of the applications of event-based systems is that all make particular, application-specific assumptions. These make it rather difficult to generalize and reuse the systems for other applications, or make it very hard to integrate event-based applications that grew separately. Here we try to identify a few areas where research is needed and could yield benefits for the whole area of event processing.

Event Algebra and Semantics.

Event algebras with their specification semantics were developed in various areas, such as active databases, temporal or deductive databases, temporal data mining, time series analysis, and distributed systems. Event rule specification has been studied in active database systems for several years with special focus on composite events and temporal conditions. Active database systems can rely on the transactional context for the composition of events. ECA conditions can be defined based on the old and new state of the database. Temporal interval operators have been used for the ordering of database states. Ordering based on events, as opposed to states, has been implemented in the SAMOS system [19]. Active database systems do not support adaptive system behavior; they are often implemented as centralized systems that deal only with database-internal events. In the context of active databases, temporal logic has been used to describe the semantics of composition operators, e.g., Enhanced Past Temporal Logic (PTL) that supports relative temporal conditions and composite actions. The areas of temporal data mining and time series analysis rely on temporal association rules - from a set of data, rules verified by the data have to be discovered. In distributed systems, events have to be ordered based on possibly incorrect timestamps. This may lead to uncertainty in event ordering. An extensive number of event-based systems have been designed and implemented. Most of those follow individual event semantics that are not, or not completely, explicitly defined. This makes comparisons or collaboration between systems chal-

Table 1: Applications and their features

	Mobile tourist IS	Baggage tracking	Avalanche warning	Info dissemination	Blog Info m & d	Fraud detection	Financial	Supply chain mgmt.	Railway cargo	Cockpit info syst.	Traffic monitoring	Health care	Ambient asst. living	Smart cities	Self-X systems	Threat detection	Gaming
Temporal events	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Absolute position events	✓	–	✓	–	–	✓	–	–	✓	✓	✓	*lim*	*lim*	✓	–	✓	✓
Logical positioning	✓	✓	✓	–	–	–	–	✓	✓	✓	✓	✓	✓	✓	–	✓	✓
Spatio-temp corr.	✓	✓	✓	✓	–	✓	–	✓	✓	✓	✓	✓	✓	✓	–	✓	✓
Other context in events	✓	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Change events	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Status events	✓	–	–	✓	–	–	✓	–	✓	✓	–	✓	✓	✓	✓	✓	✓
Interval events	✓	–	✓	–	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Event sequencing	–	✓	✓	*lim*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Homog. aggregation	–	–	✓	–	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Heter.composition/fusion	–	–	–	✓	–	–	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Event enrichment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
Derived events	✓	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Event re-use	✓	–	✓	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
Outlier handling (d/s)	s	d	d/s	–	–	d	d	d/s	d/s	d	d	d	d	d	d	d	–
Early filtering	–	–	✓	–	–	–	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
Event purging	✓	✓	–	✓	–	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–
Event persistence	✓	*lim*	*lim*	✓	✓	✓	✓	✓	✓	✓	✓	✓	*lim*	✓	✓	✓	✓
Audit trail	–	–	–	–	–	✓	✓	✓	✓	✓	–	✓	✓	–	✓	–	–
Event propagation/notif.	✓	✓	✓	✓	✓	–	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Delivery guarantees	–	✓	–	✓	–	✓	✓	✓	✓	✓	–	✓	✓	–	✓	✓	✓
False positives	✓	✓	✓	✓	–	✓	✓	–	✓	✓	✓	✓	✓	–	✓	✓	–
False negatives	–	–	✓	*lim*	–	✓	✓	–	✓	✓	–	✓	✓	–	✓	✓	–
Transaction processing	–	–	–	–	–	–	✓	✓	–	–	–	–	–	–	–	–	–
Point of processing ($d/c/n$)	d	c	d/c	d/c	c	c	c	d	d/n	d	d	d	d	d/n	d/c	d/n	d/c
Heterog. of platform	m/h	l	l	m	l	l	l	m	h	l	m	m/h	m	h	l/m	l/m	l
Volume	l/m	m	h	m	h	h	h	m/h	h	m	m/h	$l/m/h$	l	h	$l/m/h$	$l/m/h$	h
Security	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	✓
Privacy	✓	–	–	–	–	✓	✓	✓	✓	–	✓	✓	✓	✓	–	✓	–
Mobility of event source	✓	*lim*	–	*lim*	–	✓	–	✓	✓	✓	✓	✓	✓	✓	–	✓	✓
Mobility of subscriber	✓	–	✓	*lim*	✓	–	–	–	✓	✓	✓	✓	✓	✓	–	✓	✓

✓ indicates required features, *lim* a feature that is taken to a limited extent,
$d/c/n$=distributed/central/node, $l/m/h$=low/medium/high, d/s=detect/suppress

lenging. The need for a complete event algebra to describe event semantics has already been identified at the Dagstuhl Seminar [8].

Operative Aspects.

Complementing the event algebra, operative aspects must define issues, such as, the life cycle of events, their replication, consumption modes, event logging, validity intervals, and when events can be discarded or disregarded, for example, when they arrive late. While some early work on event consumption was very useful in addressing these issues, there is no common framework for expressing operational semantics of event-based systems in a consistent manner. This leads to inflexible systems and errors when new applications subscribe to existing event streams.

Event Enrichment.

This is a wide-open and difficult research area dealing with the combination of information from heterogeneous sources to produce events of higher level of abstraction. To meaningfully enrich events, metadata must be added, often automatically extracted from semi-structured data. Event enrichment calls for an understanding of the semantics not only of the events but also of the external sources of information. Depending on the degree of abstract knowledge needed, this may call for human involvement in response to automatically generated recommendations [28].

Mobility in Event-based Systems.

Mobility implies frequent disconnections and reconnections, less reliable wireless communication and the need to guarantee delivery of events under these conditions. Open issues exist on what guarantees can be given, in how to anticipate movement, how to stage events in anticipation of movement, how to persist event histories for replaying them and about the semantics and the life-cycle of events with deferred delivery.

End-to-end Security.

E2E security in event systems using dedicated communication channels is similar to any messaging system. However, when publish/subscribe notification is used, new mechanisms for delimiting the visibility of events, managing the keys for type and attribute level encryption and the use of role based access control are needed. E2E security is especially challenging in sensor networks based on low capability nodes and broadcast communication. [39].

Privacy.

Privacy in this context refers on one hand to protecting the information about subscribers, their interests (as defined in rules and profiles) as well as the information about which notifications they receive since these may be mined for patterns. On the other hand, privacy requirements also apply to event producers, especially in the health domain and when events generated by wearable or portable devices can be used for tracking. The success of many technically feasible and potentially beneficial applications may hinge on proper handling of the privacy concerns and the public perception.

Dealing with Heterogeneity in Mixed-mode Environments.

Little experience exists in integrating low end sensors with high end stream processing engines. Cyber-physical systems pose special challenges for middleware, communication substrates, and programming environments that often must be aware of continuous and discrete computation models and the highly dissimilar capabilities of the components but must mask these and must provide the appropriate mappings across platforms.

Software Engineering of Event-based Systems.

Event systems are based on a different programming paradigm. This requires new models, and, most important, a coherent software engineering framework supported by appropriate tools. Current research and development has addressed development of individual components, such as the notification mechanism or the event processor. However, for widespread acceptance a comprehensive process is needed that addresses the design, captures functional and non-functional requirements, integrates seamlessly event detection, processing, communication and responders, provides guidelines for testing and deployment of event systems and establishes the policies for administration and governance.

Performance Modeling, Capacity Planning, Benchmarking.

Industry is asking for reliable and unbiased performance modeling and evaluation. This will require the involvement of organizations, such as TPC and SPEC. STAC, an organization supported by financial institutions, is benchmarking CEP engines. SPEC has released its SPECjms2007 benchmark [44]. Academic attempts include, e.g., [4] in the area of benchmarking and [33] in the area of performance modeling.

References

[1] I.F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8), 2002.

[2] G. Buchanan and A. Hinze. A generic alerting service for digital libraries. In *5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'05)*, pages 131–140, New York, 2005.

[3] A. Buchmann. Infrastructure for smart cities: The killer application for event-based computing. In *[8]*, 2007.

[4] A. Carzaniga and A. L. Wolf. A benchmark suite for distributed publish/subscribe systems. Technical report, Department of Computer Science, University of Colorado, 2002.

[5] S. Chakravarthy and Q. Jiang. *Stream Data Processing: A Quality of Service Perspective.* Springer, 2009.

[6] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In *20th Int. Conference on Very Large Data Bases (VLDB'94)*, pages 606–617, 1994.

[7] S. Chandrasekaran and M. Franklin. Streaming queries over streaming data. In *28th Int. Conference on Very Large Data Bases (VLDB'02)*, pages 203–214, 2002.

[8] K. M. Chandy, O. Etzion, and R. von Ammon, editors. *Event Processing*, number 07191 in Dagstuhl Seminar Proceedings. IBFI, Schloss Dagstuhl, Germany, 2007.

[9] K. Mani Chandy. Event-driven applications: Costs, benefits and design approaches. Gartner Application Integration and Web Services Summit 2006, San Diego, CA, June 2006.

[10] K. Mani Chandy and M. Olson. Federated event systems: The event web. http://www.ebizq.net/topics/cep/features/9428.html, June 2008.

[11] J. Chen, D. DeWitt, F. Tian, and Y. Wang. Niagaracq: A scalable continuous query system for internet databases. In *SIGMOD International Conference on Management of Data*, pages 379–390, 2000.

[12] M. Cilia, M. Antollini, C. Bornhövd, and A. Buchmann. Dealing with heterogeneous data in pub/sub systems: The Concept-Based approach. In *International Workshop on Distributed Event-Based Systems (DEBS'04)*, Edinburgh, 2004.

[13] D. Cook and S. Das. *Smart Environments: Technology, Protocols and Applications*. Wiley, 2005.

[14] P. DeVries. The state of RFID for effective baggage tracking in the airline industry. *International Journal of Mobile Communications 2008*, 6(2):151–164, 2008.

[15] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[16] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining Knowledge Discovery*, 1(3):291–316, 1997.

[17] L. Fiege, M. Mezini, G. Mühl, and A. Carzaniga. Buchmann. Engineering event-based systems with scopes. In *16th European Conference on Object-Oriented Programming (ECOOP'02)*, pages 309–333, 2002.

[18] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: you only get one look a tutorial. In *SIGMOD International Conference on Management of Data*, pages 635–635, 2002.

[19] S. Gatziu, A. Geppert, and K. Dittrich. The SAMOS active DBMS prototype. In *SIGMOD International Conference on Management of Data*, page 480, 1995.

[20] A. Giffords and M. Palmer. Streambase real-time profit & loss white paper. `http://complexevents.com/wp-content/uploads/2008/09/streambase_whitepaper_real_time_pnl.pdf`, last accessed 16 September 2008.

[21] B. Glover and H. Bhatt. *RFID Essentials (Theory in Practice (O'Reilly))*. O'Reilly Media, Inc., 2006.

[22] C. Grothe, S. Bauer, and U. Klingauf. Overcoming the media breaks in ais: Dissemination of operational change notifications by xnotam. In *1st Int. Workshop on Aircraft System Technologies (SAT'07), Hamburg, March*, 2007.

[23] P. Guerrero, K. Sachs, M. Cilia, C. Bornhövd, and A. Buchmann. Pushing business data processing towards the periphery. In *23rd International Conference on Data Engineering (ICDE'07), Istanbul, Turkey,*, 2007.

[24] A. Gupta and I. Singh Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering Bulletin*, 18(2):3–18, 1995.

[25] T. Harrison, D. Levine, and D. Schmidt. The design and performance of a real-time CORBA event service. In *12th ACM SIGPLAN Conf. on Object-oriented programming, systems, languages, and applications (OOPSLA '97)*, pages 184–200, 1997.

[26] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. *IEEE Network*, 18:2004, 2004.

[27] A. Hinze and G. Buchanan. The challenge of creating cooperating mobile services: experiences and lessons learned. In *29th Australasian Computer Science Conference*, pages 207–215, Darlinghurst, Australia, 2006.

[28] A. Hinze, G. Buchanan, D. Jung, and A. Adams. HDLalert – a healthcare DL alerting system: from user needs to implementation. *Health Informatics Journal*, 12(2):121–135, June 2006.

[29] A. Hinze and A. Voisard. Location- and time-based information delivery in tourism. In *8th International Symposium in Spatial and Temporal Databases (SSTD)*, Santorini Island, Greece, 2003.

[30] A. Joseph, A. Beresford, and J. Bacon et al. Intelligent transportation systems. *IEEE Pervasive Computing*, 5(4):63–67, 2006.

[31] D. Jung and A. Hinze. A mobile alerting system for the support of patients with chronic conditions. In *First European Conference on Mobile Government (EURO mGOV), Brighton, UK*, pages 264–274, 2005.

[32] P. Kabus and A. Buchmann. A framework for network-agnostic multiplayer games. In *EUROSIS GAME-ON Int. Conf. on Intelligent Games and Simulation*, 2007.

[33] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann. A methodology for performance modeling of distributed event-based systems. In *11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pages 13–22, Washington, DC, USA, 2008.

[34] L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. *IEEE Trans. on Knowl. and Data Eng.*, 11(4):610–628, 1999.

[35] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston,, 2001.

[36] G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer, 2006.

[37] M.C. O'Connor. San Francisco airport OKs RFID bag-tracking pilot. *RFID Journal*, 2006.

[38] N. W. Paton, editor. *Active Rules in Database Systems*. Springer, New York, 1999.

[39] L. Pesonen, D. Eyers, and J. Bacon. Encryption-enforced access control in dynamic multi-domain publish/subscribe networks. In *Int. Conf. on Distributed Event-Based Systems (DEBS'07)*, pages 104–115, 2007.

[40] J. Reason and R. Crepaldi. Ambient intelligence for freight railroads. *IBM Journal of Research and Development*, 53(3), 2009.

[41] J. Rhyner. Avalanche warning: components of a well-established warning system. *Forum für Wissen*, 2007.

[42] H. Roitman, A. Gal, and L. Raschid. On the challenges in event delivery. Fast abstract, Int. Conf. on Distributed Event-Based Systems (DEBS'08), 2008.

[43] D. Rosenblum and A. Wolf. A design framework for internet-scale event observation and notification. *SIGSOFT Softw. Eng. Notes*, 22(6):344–360, 1997.

[44] K. Sachs, S. Kounev, J. Bacon, and A. Buchmann. Workload characterization of the SPECjms2007 benchmark. *Performance Evaluation*, 2009. in Press.

[45] Sun Microsystems, Inc. Java platform, enterprise edition (Java EE) specification, v5, May 2006.

[46] Georgia Tech. Aware home. http://awarehome.imtc.gatech.edu/, 2009.

[47] K. Terfloth, K. Hahn, and A. Voisard. On the cost of shifting event processing within wireless environments. In *[8]*, 2007.

[48] TIBCO. Tib/rendezvous. White Paper, TIBCO, Palo Alto, CA., 1999.

[49] J. Widom and S. Ceri, editors. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[50] Ian H. Witten and Eibe Frank. *Data mining : practical machine learning tools and techniques*. Morgan Kaufmann Series in Data Management Systems. Elsevier, Morgan Kaufman, 2005.

[51] M. Wu, A. Liu, and K. M. Chandy. Virtual environments for developing strategies for interdicting terrorists carrying dirty bombs. In *5th International ISCRAM Conference*, 2008.