Dagstuhl Seminar on Event Processing

May 6 – 11, 2007, Schloss Dagstuhl

Organizers: Mani Chandy, Opher Etzion, Rainer von Ammon.

Summary of the presentations was done by Peter Niblett.

During the week of May 6-11, the event processing Dagstuhl seminar took place; In this seminar there were 43 participants from the following countries: Canada, Germany, Holland, Ireland, Israel, Korea, New-Zeeland, Portugal, Spain, Sweden, Switzerland, UK, and USA. The seminar had unusual proportion of industrial participants and included participants from: CITT, Cordys, Gartner, IBM, IDS Scheer, Microsoft, SoftwareAG, Oracle, RuleCore, and WestGlobal.

The seminar also consisted of people with several core disciplines such as: distributing computing, databases, software engineering, business process management, sensor networks, simulation and verification.

One of the participants commented that it seems that both academic and industrial people are interested in languages and implementation issues of complex event processing (either by rules or queries), in addition the academic people only were interested in the fundamental middleware issues (maybe since the industry people view it as engineering topics and not as research topic), while the industry people were interested on locating event processing in the buzzword-oriented universe (SOA, BI, BAM), a discussion that lacks research content, in the view of the academic people. Everybody though that the participants of the industry people had major contribution to this seminar. At the concluding session, the industry people compiled a list of research topics that the industry wishes to see, and the research people produced their wish list from the industry.

The seminar held deliberations (with some evening sessions in the wine cellar) on: what is event processing – use cases and classifications, is it a paradigm shift? Positioning EP relative to industry buzzwords (SOA, BAM, BI…), Semantics issues, modeling issues, and implementation issues.

Dagstuhl Seminar Proceedings 07191
Event Processing
http://drops.dagstuhl.de/opus/volltexte/2007/1148

The First Day: Monday

Session 0: Topics for the week

The first session discussed issues for inclusion in the week's agenda:
1. SOA and EDA overlap loose coupling, relationship to EAI
2. Rule management, Evolution of rules. Visualization of rules
3. Streams and Event management
4. Scalability of all aspects (very high volumes, number of rules, number of recipients etc)
5. Operational characteristics, "ilities", system guarantees, security
6. Approaches to distributed event brokering, aspects, timing
7. Benchmarks, good benchmarks, design points
8. Differing models for event processing, what is appropriate to what applications. Also models for events and modeling notation
9. Semantics of events, temporal uncertainty, programmatic semantics
10. Characteristic application space for EP - scenarios, current and future
11. Relationship between Events and BI. "Event pattern mining"
12. Spam and false positives
13. Vertical markets
14. Standards - what do we need?
15. Event acquisition. Control of sensor networks, Identification of time-critical events

Three Meta issues from Roy Schulte
a) What areas should research focus on?
b) What should industry do?
c) Is this a Paradigm shift?

Assignment of these topics to agenda sessions:

- Taxonomy/Classification of event applications (Monday)
- Models, semantics (Tuesday)
- Programming models (Tuesday)
- Implementation: Performance, Benchmarks, "ilities" (Wed/Thu)

Session 1: Taxonomy of event applications

Opher Etzion started the discussion with a description of 5 classifications of application:

- Event processing as part of Business Logic: In this class of application the event system needs the same reliability / transactionality and other qualities that the business logic has.
- Event processing as Business Observation. In this class, the events are not actually part of the business logic, but sit alongside it and are used for

monitoring, management and compliance. Examples include Business Activity Monitoring (BAM) and fraud detection.

- Event Processing as part of Information Delivery. In this class of application the events are primarily used to stream information, such as stock prices and news feeds. Many traditional publish/subscribe applications fall into this category.
- Event Processing as part of Problem Determination. Events are generated when a problem symptom is detected. Applications in this class may then need to do some retrospective analysis to determine the actual problem cause.
- Event Processing as part of Prediction Systems. In this class raw events are examined (and possibly compared with historical data) in order to predict future situations of interest.

There was a brief discussion of what was special about Event Processing as opposed to other approaches, and whether it was possible/useful to have a general purpose platform and tooling for Event Processing. To help shed some light on this, the meeting then discussed 22 scenarios:

1. **Airline baggage (Opher Etzion)**; Survey shows that 1.5% of all airline bags don't arrive with the passenger, and 15% of those are never delivered. This scenario involves use of RFID and event processing to detect situations that lead to bag loss and also help track bags when they are lost.
2. **Patient health monitoring (Opher Etzion, Annika Hinze);** Many health monitoring systems "fail safe" and generate more alerts than are required (i.e. many false positives). These results in information overload for the healthcare workers and consequently the workers run the risk of ignoring something important. This scenario involves correlating events from multiple monitors and also applying knowledge of the individual patient's condition in order to filter out as many false positives as possible. The scenario was extended by Annika to include input from multiple stakeholders (patient, nurse, doctor), and the language/tools used needs to be able to accommodate these differing users.
3. **Rose bowl (Mani Chandy);** Protection against terrorist threat against a major sporting event. The threat is a suicide bomber exploding a radioactive "dirty bomb". A physical check at entry to the arena is impractical, so instead the approach is to have agents with Geiger counters patrolling the crowd. Event processing is used to correlate data returned from these detectors. This scenario illustrates the trade-off between false positives and the catastrophic effect of a false negative.
4. **Particle Physics (Tore Risch);** Use Event processing to analyze the results from CERN as part of the search for the Higgs Boson. The raw events are analyzed to detect information about the types of particles, their positions and speeds. There is a huge amount of data to be processed, so processing is triaged into several steps (Cuts). The order in which these Cuts are performed is very significant (a good order can be over 1000x better than a poor order). The project involved use of SQL queries to do the analysis, as opposed to using bespoke C++ filters.
5. **Radio astronomy (Tore Risch);** aggregating data from 13000 radio detectors. Each detector produces 100 Million Events per second, giving a total of 30 TB/s of raw data. It is not feasible to store all this data and then mine it in a conventional fashion. There is so much data that it has to be processed as it comes in; it is impractical to store it first. Also it is impractical to do all the processing

for each raw event, so the processing to be done is conditioned by previous events that were detected.

6. **Traffic monitoring (Sharma Chakravarthy);** a traffic monitoring system in California gathers GPS data transmitted by cars (Car Id, position, and speed). Each car transmits a packet of data every 20 seconds. The monitoring system uses Event Processing to correlate this data and detect traffic problems, routing alerts upstream and downstream from the actual problem.
7. **Spread of infectious diseases (Sharma Chakravarthy);** Use of Event processing to predict the spread of viral infections across a large country, such as the United States. The raw data is already gathered and is held by the US Center for Disease Control, but it isn't really being used to predict disease spread at present.
8. **Monitoring of queries (Vijay Dialani);** Continuous queries are used to monitor events. Queries by themselves can be queried upon and help provide the richer context. Queries could be mined to discover patterns of interest, and make inferences about preferences of the users/applications. This blurs the line between data and queries and also imparts adaptivity to queries.
9. **Intelligent matchmaking (Vijay Dialani);** Potential publishers and consumers of events (for example potential buyers and sellers of a stock) register with a service that does smart matchmaking. Smart matchmaking includes detecting the patterns of events that actually occur so that events are routed to the optimal consumers.
10. **Mobile Location-based services (Annika Hinze);** Use of Event processing in conjunction with services delivered to mobile users. There are several types:
    a) Travel information systems. A mobile user travels around a country and the idea is to deliver information that is relevant to the location that he or she is in.
    b) Attendance at events. In this context "event" means something that someone attends, such as a meeting or concert (there may be hundreds of attendees). This time the idea is to deliver information that is relevant to the event(s) that he or she is attending. Note that this does not necessarily involve a particular geographical location, as we can have events that take place in multiple physical locations simultaneously.
    c) Data collection. Gathering of data from mobile users, e.g. Nature surveys (counting trees or Kiwis), farming (recording yield from a particular sheep after shearing).
    d) Traffic data. Information about plane or train data is streamed to an event process. Needs to be processed with a read-out of other data relative to the local area.
11. **Physical Intrusion detection via sensors (Kirsten Terfloth);** Use of ten sensors attached to a fence to detect a physical intruder. The system has to distinguish between someone climbing over the fence (positive result) from negative ones (e.g. someone banging into the fence) this decision may need to take into account data from multiple sensors. Challenges include limited CPU in each device. There is a design trade-off question, as to whether it is better to filter in the devices or in the network.
12. **Avalanche prediction (Katharina Hahn);** as with the intrusion detection this scenario involves multiple sensors. They are based at different places on the mountain-side, and the prediction needs to take into account several influential parameters. Some of these parameters are static, for example the steepness of the mountain at the place where the sensor is positioned. Other parameters are

dynamic, for example temperature, air pressure and snow depth. It is not just the current value of these dynamic parameters that matters, the history is also important. The question studied was to find the most cost-efficient deployment, whether it is better to have lightweight sensors streaming raw data to a base station, or to do more processing in the sensors themselves. Where the historical data should be kept? Factors to consider include the cost and reliability of the sensors, and the power that they consume, against the data transmission costs and latency requirements of the application.

13. **Use of EP as an alternative to Request/Reply in software engineering (Claudi Paniagua Macia);** The conventional request/reply paradigm can be recast into Event Processing terms. This will make software design more adaptable and flexible and will avoid problems caused by statefull services. Current OO approaches are too low-level, exposing things like threads and instance variables.
14. **Business Activity Monitoring (Tobias Blickle);** A business process (written in BPEL or similar) is instrumented to generate events as it transitions between steps and these events are analyzed in order to drive a dashboard. The dashboard can offer a number of perspectives
    a) Exceptional situations
    b) Historical process performance
    c) Track and Trace
    d) Organizational view

    Events flow into the monitoring system which then tracks status (for example KPIs). Some of the perspectives (e.g. the Exceptional situations) involve generation of alert events based on observation of this status. The rules used to define these exceptional conditions need to be specifiable by business analysts. Timeliness of information and alerts is critical.
15. **Governance, Risk and Compliance (Tobias Blickle);** this scenario is about compliance applied to Business Process. Monitoring is performed in a manner similar to scenario 14, except that now the system has to perform pro-active detection rather than retrospective detection. One use of this is to detect fraud.
16. **Smart City "Ambient Intelligence" (Alex Buchmann);** providing seamless event processing capabilities across spaces and domains is one of the main challenges of Ambient Intelligence. The vision of AmI calls for the vanishing of the computational infrastructure and pervasively available, context-aware services. This in turn requires from the infrastructure to provide 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-X 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.

17. **Detection of money-laundering (Dean Jacobs);** Banks now have a regulatory requirement to install money-laundering detection software. This looks for unusual transfers of funds (e.g. an atypically large amount of money moving into an account and then straight out again). Each alert causes significant cost to the bank as it has to be investigated by humans, and with a false negative there is the possibility of annoying an innocent customer. Banks are really only concerned about meeting regulatory requirements. There is a high amount of data (every transaction has to be examined) but speed of detection is not critical.
18. **Tailored internet search across heterogeneous providers (Dean Jacobs);** Internet users currently use a variety of information sources, e.g. Google, del.icio.us, Yahoo each with different search languages. In this scenario we use EP to analyze peoples' searches so as to get a view of their interests and preferences to give them a more tailored experience. The issue here is the mixed nature of the information sources which have to be adapted - we assume that these sources are not modified.
19. **Supply chain visibility (Henry Chang);** this scenario concerns optimal management of a physical supply chain. Events that affect the chain (e.g. purchases) are captured and analyzed with reference to three types of data: Historical Supply Data, Historical Demand Data and Reference Data (e.g. part-number catalogs). The aim of the analysis is to detect possible problems (e.g. shortages) and allow the manager to take steps to condition supply and/or demand. Some of the events are periodic (e.g. end-of-day stock check) and others a-periodic (exceptional conditions such as a large order cancellation). The number of rules is quite small (about 10) and most of the complexity is in deriving KPIs. An automated system reduces the amount of human involvement required. Timeliness is not absolutely critical.
20. **Algorithmic trading**; Well-known application of event processing. It is worth noting that major Wall Street firms have had bespoke C++ implementations for some time.
21. **Fraud detection;** Similar to money-laundering, except that this time the banks have a very clear interest in detecting as many frauds as possible and as quickly as possible.
22. **Massive Multiplayer Online Games;** These games typically have proprietary event-driven architectures that give a shared playing environment to 10s or 100s of users (out of a total of 100,000+ players). Throughput is not excessive, but responsiveness is critical.

Classification questions

1. Which "ilities" are significant?

a) Scalability - Ultra High throughput, Scenarios 4, 5, 20
- High throughput 8, 9,10,13,18, 19, 22
- Medium throughput 1, 15

b) Recoverability - Important 1,2,11,15,20,22

c) Privacy 10

d) Security 17

e) Cheat detection 22

2. What requirements on Timeliness? Is it sub-second?

Order of 10 seconds or more: 1,4,5,7,8,12,14,15,17,19

Order of 1 second: 3, 10, 11
Sub-second 20, 21
[Algorithmic trading 1ms is equivalent to $100m per year]

2a. is this scenario distributed in nature?
Yes: 1, 4, 5, 6, 9, 11, 20

3. What value-add does this solution provide?
End-user productivity 2.3,8,9,10,11,12,14,19,20
Application was not done before 4.5,6,17,21,22

4. What is the complexity of the detection logic?
Very High 3, 4,5,6,7,12,13,19
High 1,11,17,21
Medium 8, 9, 10, 14

5. What is the relative cost of false positives and negatives?
High 2, 3, 7, 11, 12, 15 (sometimes)
Low 19

6. What amount of work is required to integrate EP with existing systems?
High 1, 2, 7, 11, 14, 15, 17, 19, 21
Medium 3, 10 (building specific, can GPS be used)

7. What is the special value-add of Event Processing (if any)?
This question was not addressed

This analysis is summarized in the following table:

	Through-put	Recover-ability	Security Integrity Privacy	Latency	Distributed	Value add	Complexity of detection	Cost of false +ves
Patient health monitoring		Needed				Productivity	Low	High
Mobile Location-based svcs	High		Needed	1		Productivity	Medium	
Monitoring of queries	High			10+			Medium	
Business Activity Monitoring				10+		Productivity	Medium	
Business Process Compliance	Medium	Needed		10+			Medium	High (some)
Intelligent matchmaking	High				Yes	Productivity	Medium	
Tailored internet search	High						Medium	
Fraud detection				0.1		New app	High	
Physical intrusion detection		Needed		1	Yes	Productivity	High	High

Smart City (Ambient intelligence)	High	Needed	Needed	1	Yes	New app	High	High
Airline baggage	Medium	Needed		10+	Yes		High	
Detection of money laundering			Needed	10+		New app	High	
Multiplayer games (MMORGs)	High	Needed	Needed			New app	High	
"Rosebowl" security				1		Productivity	Very High	High
Infectious disease				10+			Very High	High
Avalanche prediction				10+		Productivity	Very High	High
Supply chain visibility	High			10+		Productivity	Very High	Low
Traffic monitoring					Yes	New app	Very High	
Algorithmic trading	Ultra	Needed		0.1	Yes	Productivity	Very High	
Particle physics	Ultra			10+	Yes	New app	Very High	
Radio astronomy	Ultra			10+	Yes	New app	Very High	

Monday Evening Session. Will we see an EP/EDA paradigm shift?

Roy Schulte posed the question "Will Event Processing (EDA) become a paradigm shift in the next few years or not?"
Examples of technologies that have resulted in Paradigm Shifts include Databases, GUIs, and OLTP. Examples that haven't seen the widespread general adoption that would be entailed by a paradigm shift include Artificial Intelligence, Distributed Objects/CORBA, and Open Systems. We have seen things like EDA before (e.g. Message Driven Processing) - so is the time now right for EDA?
Points made during the discussion included:

1. We need to decide whether we are talking about Asynchronous Processing, e.g. triggering transitions in business applications via events, or full Complex Event Processing. The answer might be different for the two. Some people thought that an Asynchronous Programming paradigm shift is already underway.

2. Paradigm shifts can't happen if there are too many barriers. Historically the barriers to EP/EDA have included:
    a) The "flow-chart" style synchronous programming is easier to teach, learn and debug.

b) The non-deterministic (anarchic?) nature of EDA does not necessarily make it attractive as a way to design repeatable business processes.
c) Variety of different proprietary tools and languages. It is hard for non-technical people to use them.
d) General purpose EDA platforms don't have enough function, so developing applications is costly (people have to code a lot of additions as part of their applications)
e) Lack of awareness of EDA in the business, analyst and programming communities

3. Paradigm shifts are more likely to happen when adopters decide they need a whole new avenue of applications; they are less likely to happen as a way of re-engineering existing systems. New applications might include automated health care (for example the German population will reach 1:2 old: young ratio by 2020 so today's model of health care will not be viable), and intelligent cities.

4. Paradigm shifts usually happen as a result of some external change, not just because of innate strengths of the technology itself. Possible changes that might help EP/EDA include
    a) Emergence of machine-born data. In the "old days" data had to be entered into computer systems by humans. Now there is much greater deployment of sensors, RFID readers etc. and therefore much larger amounts of data available. It isn't economical to store it all in static databases and write traditional database-oriented applications.
    b) Multi-core processors will force a rethink of traditional application design. The concurrency issues with conventional request/reply applications make it hard to exploit 64-way machines effectively
    c) Massively Multiplayer Online Realtime Games are training a new generation of programmers to think in an asynchronous, event-driven way.
    d) Growth in computer-computer interactions (as opposed to human-computer interactions)
    e) Convergence of physical and digital worlds

5. Standardization is not necessary for a paradigm shift, but good, appropriate standards (de facto or otherwise) certainly help

Conclusion (Roy). There will be a paradigm shift, but it might not be called Event Processing.

The Second Day: Tuesday

Session 2 – Semantics

1. **Opher Etzion** gave an introductory talk outlining various areas of the subject
    a) Event Semantics. This is concerned with
        i. What is an event?
        ii. What general properties do events have?
        iii. What relationships are there between events and between events and other entities?
        iv. What do we need to specify about event context?
    b) EP Network Semantics or semantics in the large. This includes
        v. the nodes and edges in the network
        vi. Is context a distinct semantic entity?
    c) EP Agent Semantics, semantics in the small.
        vii. specific nodes in the network

2. **Graph Transformation approach to EP (Claudi Paniagua Macia)**; **Claudi** proposed that Event Processing can be viewed as graph transformation. If you take a partially ordered set of events and view them as a graph, then an Event Processing agent is in effect transforming that graph (by executing a set of transformation rules) into another graph. This transformation involves adding or removing nodes from the graph. The Event Processing agent may need to keep state in order to do this transformation.
   He then went on to illustrate with an example where he took a conventional request/reply message exchange (push and pop operations on a stack) and analyzed it as 8 events (1 event for client sending push request, 1 for server receiving push request, 1 for server sending push response, 1 for client receiving push response; another 4 events for the pop operation). The graph containing these events could be transformed into a graph containing complex events derived from these raw events, for example the 4 raw push events could we combined into a single complex Push event. One application would be for a monitoring application to check operational correctness; a graph collected from runtime operation could be analyzed to check that the various types of events were correctly related (e.g. pop responses contained expected values).
   Francois Bry commented that this kind of verification was not novel. Claudi responded that verification was not the goal, just an example or test of how graph transformation could be used.

3. **Analysis of the new SQL extensions (Carlo Zaniolo); Carlo's** presentation covered the EP-related extensions recently added to SQL by Oracle, IBM, Streambase and others. He focused on the Regular Expression based pattern matching capability. When you are executing a SQL expression over a stream of events, this new Regular Expression capability can be used as a way of defining certain complex events. Carlo's impression was that the new function was very expressive and easy to understand.

There have been previous attempts to do event pattern matching languages, what is new this time is its integration with SQL, so you get the same query language for pushing and pulling events. Also SQL query optimizations are possible and it is possible to do more advanced semantics, like Snoop's chronicle context.
Opher commented this was useful but that there were other more expressive languages and we shouldn't assume that all CEP languages had to be extensions of SQL.

4. **EP models in high-energy physics (Tore Risch); Tore** went into greater depth on the CERN application. The application has a large volume of data, organized into reasonably large events. Each event is subject to a battery of filtering tests to identify the types of particles involved and determine their positions and velocities. These filters, referred to as Cuts, operate on one event at a time and are written by reluctant C++ programmers. They can be nested and get quite complex. Tore's project had examined two questions
    a. Could this be done using a Query Language rather than C++ (the advantage of a Query language would be its flexibility and adaptability)
    b. How to determine a good order to run the various cuts (experience from the C++ code was that the order could affect performance by 1000x)

    They were able to use an SQL-like query language, but found that traditional cost-model based optimization approaches didn't work - the data was too complex. In fact choosing a random order of cuts often worked quite well. The approach they took was to try random orders across a large body of sample data and measure performance so as to find a good ordering.

5. **Snoop and its semantics (Sharma Chakravarthy); Sharma** gave a review of the Event Condition Action paradigm from its origins in 1980s applications, through Object Oriented systems in the 1990s, such as HiPAC. The important step was the separation of event from condition (previously the difference had been blurred). ECA had been incorporated into relational databases as database triggers, but these had not been widely used. A general discussion on database triggers ensued, observations were that they weren't supported the same way by all vendors, their semantics were unhelpful, there was poor guidance to users, they had performance problems and they didn't scale up - the conclusion was that people shouldn't attempt to use them.
    Sharma then gave a brief taxonomy of events. He divided them into Primitive events (point in time) and Complex/Composite (interval-based). He then outlined the Snoop expression language and its operators. This can be used to create composite events out of other events. Simple use of operators can yield too many composite events, so the language also includes a Context concept. The Context can be used to specify conditions on when composite events are generated. E.g. a Fixed or Sliding window.
    Opher commented that HiPac and Snoop deserved credit as being the ancestors of modern EP work, but asked why it was that Active databases hadn't been successful in the 1990s, and wondered if things were different now. His observations were that the industry wasn't ready then, and the close association with the database (and triggers) was a problem.

6. **Filters and Composite Events (Susan Urban); Susan** took the ECA model used by Snoop and suggested that some of the composition aspects of a CEP rule

could be pushed down into the Filtering layer, and treated as Filters. The idea would be to define a Composite Event declaratively and let the filter (network) layer produce these events - they could then be subject to more complex processing by rules in the conventional fashion. The advantage of this approach is that it could do more aggressive filtering of data, and would reduce the generation of unneeded events (if no-one subscribes to a particular complex event then the "rule" to generate it never gets run). Her project had successfully implemented a subset of Snoop operators (SUM, MIN, MAX, COUNT, and AVG). There was some discussion of whether something that computes a new event should be called a Filter or not. Susan argued that the name was correct, as it resulted in the underlying raw messages disappearing from the stream.

7. **Stream semantics (Jonathan Goldstein);** Conventional window-based stream processing semantics have problems when system overloads occur or when recipients require data in order. If you are computing a function that requires a certain amount of data, your options are to a) wait for all the events to arrive (high latency) or b) provide a partial answer quickly (low consistency). The principal idea was to provide a third option (medium consistency) in which the processor generates an answer quickly, and if further data arrives which modifies the result it then generates a "retraction" followed by a new answer. Since "answers" are interval-based events, the retraction need only cover a portion of the interval of the original answer. This is appropriate for many applications, since in many real-world scenarios applications have to be able to cope with business-level retractions anyway. The database community has conflated window size with the time someone is prepared to wait for an answer. This approach separates these concerns.

8. **Classification of Uncertainly (Avi Gal);** Uncertainty exists in many EP applications. An event is subject to:
    a. Uncertainty as to whether a given event actually occurred or not, for example imprecision in a sensor detecting a physical event such as temperature threshold crossing
    b. Uncertainty regarding the accuracy of attributes of an event
    c. Uncertainty of the origin of an event

    Composite (derived) events are subject both to uncertainties in the ingredient events and, in some cases, to uncertainties in the inferencing rules. For example a CEP rule to detect illegal trading can yield false positives or false negatives, i.e. Uncertainty over Occurrence. Event Processing needs to allow for this uncertainty, and in some applications it is appropriate to assign uncertainty probabilities - possibly even carrying them in event instance messages.

9. **Closing comments (Opher Etzion); Opher** proposed semantics for events, including metadata (such as source, timepoint/interval, spatial coordinates, and uncertainty). Defining the timepoint of a complex event has some challenges. He described an Event Processing network being made up of Nodes and Channels and presented a list of possible Event Processing node types. There was some discussion of the distinction between Translator and Aggregator nodes - are they distinguished from each other by state holding or by the cardinalities of their inputs/outputs?

A concept such as Context is needed to partition an event cloud, either by time, space or some other property.

Alex Buchmann asked how/when events are "retired" i.e. disappear from the cloud. **Opher** suggested it was a property of the channel. **Peter Niblett** questioned whether EP systems had to provide explicit "delete" operations, or whether it was possible to avoid them. Experience with distributed OO systems suggested it would be good if we could avoid having explicit delete. He suggested that the question of event lifecycles was a topic for further discussion - possibly appropriate as a research topic.

Francois Bry commented that there was a very wide range of EP applications and that finding a common set of semantics would be difficult - much harder than for static database queries.

Session 3 - Modeling

1. **Methods and Tools (Mikael Berdtsson); Mikael** observed that there was a pressing need to develop methods and tools for EP programming, otherwise the subject would not progress, as happened to Active databases in the 1990s. He asked a number of questions:
    a) Do we go for an implicit modeling approach, where you use traditional modeling diagrams and simply infer rules from them, or instead go for an explicit approach where events and rules are modeled as first class things? If we go for the explicit approach, what kind of UML diagrams do we extend?
    b) Do we need a Rule Markup Language?
    c) What guidance do we give an engineer as to when to use rules, and when not to use them?

Audience discussion:

Harald Schoning: Are we at a position where we can answer these questions?

Mikael: No

Antonio Carzaniga: Why do we need to give advice? Why not let people choose the most appropriate approach?

Mikael: They are asking for help. Of course it's up to them in the end, but we should give advice

Tore Risch: If it's a simple database query or if they want to use a stored procedure then they don't need events/triggers. But now suppose they have a master database and several people interested in observing changes to it. That's a case where triggers are appropriate.

Alex Buchmann: People have been trapped into thinking that event processing requires rules. You can do simple event processing without rules; events can be consumed by applications, not just by rules. We should decouple the idea of events from the idea of rules.

2. **Business Activity Monitoring (Tobias Blickle); Tobias** discussed requirements for modeling from the BAM perspective; we need to consider how to model rules and events in a way that makes sense for Business users, not just software engineers. For example in an IT helpdesk dashboard the users understand things like Exceptions and KPIs, they need to be able to specify rules in a way they can understand.

    Tobias provided a Process view example. The user establishes a KPI, say the

average time taken to handle a customer call, and wishes to track when that KPI exceeds a threshold. Another example is where the user wishes to detect if structural requirements are being violated. Today the user has to program such rules in a programming language. This may be Visual Basic, but it is still a programming language and not really sufficiently abstract to be given to customers. Processes themselves are modeled using EPC (Event Process Change).

We need to decide what kinds of things to model, and if there is a large number of rules how we can express them in way that users can understand the rules and all their interactions.

3. **WS-Topics (Peter Niblett); Peter** reviewed the OASIS WS-Topics specification, as a way of grouping different kinds of event, and classifying them. It is part of the WS-Notification standard and can be used when formulating subscriptions, or when advertising ones ability to emit certain kinds of events. It is assumed that event types can be described via XML schema (though the event instances themselves do not have to be represented as XML messages). The relationship between Topics and types of event isn't necessarily 1-1, a particular event type can be classified under more than one Topic, and a given Topic can contain multiple event types. Also a Topic can contain an XPATH predicate to constrain an event instance more precisely than the XML Schema type can.

4. **Event Algebra (Annika Hinze); Annika** proposed an abstract event composition meta-language that could describe any of today's composition languages. The idea would be not to displace these languages but rather produce a modeling language into which all the existing languages could map without losing precision. This required understanding the detailed semantics of all these languages, and making sure that meta-language was rich enough to express all the different ways these languages handled things. She has defined such a meta-language and is now working on a Meta Event Processing Network definition. There was much interest and discussion. Two of the points raised
    a) In a nested expression, should attributes of the top level be inherited by the sub-expressions?
    b) The usefulness of the system would be improved if it didn't assume a global clock.

5. **Business Performance Observation Model (Henry Chang); Henry** described a model for monitoring and dynamically managing business processes. A manager wants to make sure that everything is running ok, and also achieve continual process improvement. Events are used in two places. Firstly events from the business process system itself (either state changes or current status summary events) are monitored and used to update the monitoring context. This contains a set of metrics (things like KPIs are included here). XPATH 2.0-based maps are used to define this mapping of events to metrics. This mapping can include current metric state as well as the incoming event data. There is a special kind of Boolean metric to represent a Situation. A Situation triggers a Decision Map which decides how to react to the situation. Status reporting (e.g. Dashboards) report on the values of metrics and also list significant events that have happened (what is counted as "significant" can depend on the current state). Relationships are modeled in UML. Henry suggested that Event specifications should also include metric definitions

The Third Day: Wednesday

Continuation of Session 3 – Modeling

6. **Challenges in forming meaningful rules (Vijay Dialani); Vijay** outlined 3 challenges:
    a) Are the rules observing meaningful data? When designing rules you want them to exclude outliers from the input data, but how do you distinguish interesting data from outliers? When formulating a rule you often have some preconception of what the data is like and you design the rule using that assumption about the data. If the actual data fits that assumption then everything is all right, but if it doesn't then static rules may not produce useful results - a more adaptive rule approach is required.
    b) How can we tell whether the rules are sufficient to warrant the results they produce?
    c) How can we find new rules? Classical data mining provides ways of examining a data set to discover common patterns. Can we apply a Markov data cube approach to Event Streams?

    Opher commented that you cannot infer causality just because you detect a correlation in data. Would this approach allow you do detect causality?
    Vijay replied that just looking at the cube would not tell you about causality, but you could also look at the timing of events.

7. **Filters and Composite Events (Susan Urban); Susan** discussed the modeling aspects arising from her Tuesday talk (which described pushing some event processing down into the event definition). She argued that Event Processing should be modeled using a higher level of abstraction than SQL expressions (or other query languages) provide. She posed two questions
    a) How much of the Filter should be pushed down into the Event definition or Stream query? Things like the SEQ operator probably should not be
    b) What is the separation between event filters and rule conditions?

    She outlined two desirable features of rules
    - Termination. Where rules are allowed to call other rules you get the possibility of infinite cycles
    - Determinism (confluence)

    These issues still exist when processing is pushed down into the Event definition. Susan then outlined 3 challenges:
        i. How to create meaningful composite events? We need to be able to involve domain experts, but this is not sufficient. We also need to mine historical events in order to formulate new composite events (rules)
        ii. Assuring termination and confluence
        iii. Developing useful and meaningful dashboards

Finally she commented on the paradigm shift question. In her opinion the real shift would be the emergence of pro-active, reactive and interactive computation. Event Processing will be an enabling technology for this shift.
Alex Buchmann commented that Termination and Confluence problems occur if you allow rules to trigger other rules, and can be avoided if we limit actions so that they just operate on incoming event streams (i.e. if you just do monitoring not activation).
Susan replied that this might be an interesting research area. Termination and confluence had been problems with active databases, so might be issues here as well.
David Luckham and Peter Niblett expanded on Susan's challenge a). There are three parts to the problem

- Is the "language" used rich enough to be able to express all the kinds of composite events/rules we need?
- How can we make the composite event definitions (rules) adaptive? In some applications (for example fraud detection) the system needs to be able to react to changing circumstances and adapt the rules. Can this be done if we embed the rules inside composite event definitions? A particularly interesting case is intrusion detection, where you might want to allow intruders a certain amount of access for a limited time in order to gather evidence about them before blocking them.
- How do we best involve domain experts in the definition of rules? For example in the Patient Health Monitoring (scenario 2 from the Monday session) we need to involve patients, nurses and doctors. What kinds of user interface are appropriate?

8. Classification of Uncertainly (Avi Gal)

Avi discussed the modeling aspects arising from his Tuesday talk. He proposed the following research goal "A language and execution model for the inference of uncertain events in active systems".
Avi then reviewed some work in progress to apply Bayesian networks and probabilistic queries to unreliable/noisy sensor networks. We start with the definition of an Event History as a time ordered set of Events, bounded by a start and stop time. The uncertainty surrounding the events means that we cannot be certain whether the events actually occurred, and if they did occur we can be certain when they occurred or whether the reported attributes of the events are accurate or not. So instead of considering just one event history we are faced with a set of possible event histories. We refer to each of these as a "Possible World", each of which can be assigned a probability. The problem is that as time progresses the number of possible worlds increases rapidly. To ease the computational problem that this poses, we instead assign a Random Variable to represent the uncertainty associated with each specific event, and then represent the Event History as the set of the RVs of all the Events that could occur. We then build a Bayesian network, however this is also computationally challenging. The basic idea is to take each rule and mark it up with a "with probability p" factor to be applied in event histories where the condition holds
Avi also outlined the issues that applied to this probabilistic inferencing:

- Indeterminism (confluence). This can be resolved by defining an order of rule evaluation;

- Termination Can be solved by allowing each rule to be triggered only once
- Correctness. How can you be sure that the overall probability space adheres to specified semantics? This is done by adding semantics to the Bayesian network
- Efficiency. How can you maintain efficiency given that the Bayesian network is being constantly updated?

Avi discussed two approaches to the efficiency question. One is to do smart updating of the network, the other is to use a Monte Carlo sampling technique that bypasses construction of the network.

Jonathan Goldstein commented that the non-deterministic sampling approach was interesting; he wondered how it would work with Join and other operators. He thought that you wouldn't be able to use a Monte Carlo approach with a Top-k aggregation.

David Luckham said he could see an application in the area of fraud detection. He asked how quickly probabilities declined to a point where the event could be ignored. Avi said this would partly depend on the input event probabilities.

Francois Bry observed that causality should be built into the system, since this would help with probabilities. For example we might know that a given sensor always sent a pair of events (the original event and a confirmation) and this knowledge should be used - Bayesian networks aren't an appropriate way to deal with that kind of knowledge.

9. **Applications of the Snoop framework (Sharma Chakravarthy); Sharma** had looked at a number of novel applications and to see if they could be handled by the Snoop framework. The application scenarios were
    a) Clean room manufacturing. The University of Texas has a manufacturing facility for teaching purposes. However technology change means that this has to be replaced every 5 years, which is very costly. The idea here was to do a software simulation instead of buying real machines. This simulation had worked well.
    b) Distributed plans. In this scenario military plans were held in 3 databases, but the plans needed to be changed once the military operation was underway due to some external circumstance, for example a change in the weather, and these changes needed to be propagated across the databases. This scenario required flexible transactions, and the ability to monitor both the user data in the database and system data such as utilization. Snoop was used to add additional transaction semantics that were needed (for example nested transactions) to the database. To do this they intercepted things that happened in the database (begin transaction, commit transaction, acquire lock. release lock etc) and turned them into Events that were handled by the Snoop framework.
    c) Traffic monitoring. The application here is to look at incoming traffic data and trying to spot accidents. One marker for an accident might be a car becoming immobile and other cars in the same segment reducing speed by more than 30%. They had tried using just Continuous Query, but found that it was better to use it in combination with CEP style event processing. To do this you split the processing into three stages, with a reduction of data at each stage. The first stage uses Continuous Query style Stream Processing to look at the incoming streams of data and generate events

when it detects significant situations - in this case when it detects a car that has stayed in the same road segment for 2 minutes, or a car that has reduced speed by more than 30% in 2 minutes. The second stage takes these events and uses Snoop-style ECA to detect complex events (in this case "accident happened"). The third stage is then to use Rules processing to determine an appropriate action.

d) Hospital Hygiene. The application here is to enforce hospital hygiene policies by tracking the movements of hospital workers and controlling door locks. For example the hospital might want to prevent a worker moving from an infectious disease ward to a maternity ward without first passing through a disinfection station. They found that Snoop could not handle this application without some extensions, for example to emit events part-way through the detection of a composite event. Opher thought that you would need to add attributes to events and this went beyond traditional ECA.

Session 4 – Implementation

1. Introduction (Alex Buchmann); Alex outlined four things that implementations had to contend with:

a. Dealing with heterogeneous Events
b. Event delivery and composition in a (multi-server) distributed system
c. Transactional semantics - in traditional transactions systems there is a central server that manages transactions, in Event Processing the consumer of the events needs to control the transaction and mediations need to be involved as well.
d. Management scopes

He then discussed the last 3 of these

- Distributed systems:
  There are problems of time stamping, ordering and clock synchronization in distributed systems. These things are important to applications, but in a distributed system there is always a degree of uncertainty. For example we cannot guarantee both timeliness and absolutely correct ordering of messages. In many cases these things can be resolved at the application level, and the problem when designing generic middleware is that the middleware cannot absorb the semantics of all its applications - so how do we make the split between the application and the middleware? Alex proposed the dictum that "the middleware must not lie" and that it should leave the application to sort things out in a way that is suitable to the given application. If we know that there is a guaranteed upper bound on latency in our network then things become easier - we can use a "2-g precedence" approach. However we can't make such latency assumptions about large-scale networks like the Internet. The best you can do in these scenarios is to ensure that two events from the same source stay in the same order. You can also have a global time service which sends out "sweeper" heartbeat events to all participants. To handle different kinds of deployment, then we need a more flexible kind of timestamp. Alex proposed a Timestamp object that had before () and after () methods on it.

The implementation of such a timestamp could then be different, depending on the accuracy of the clocks available in the system.

1. If the system has a single clock, then implementing before() and after() is easy
2. If the system has 2-g precedence then again the Timestamp can implement before() and after() without difficulty
3. If there is known accuracy interval then the methods can return a definite answer in some cases, but if the time is unstable then the method have to return indeterminacy.

- Transactional Semantics

  In publish/subscribe systems the producers and consumers of events aren't directly connected, but instead are decoupled by mediation. In many cases we wish to control the transaction used to consume the event from the consumer itself (in contrast to a normal object request where the transaction is generally controlled by the issuer of the request). So we need to extend the atomicity sphere to include the consumer. There are several possible transaction patterns, formed from combinations of the following:

  a. The Visibility of the Event. The event could become visible to consumers as soon as it has been published, only when the producer has committed its transaction, only when the producer has aborted its transaction, or "deferred" which means when the producer starts to commit its transaction.
  b. Propagation of the publisher's transaction. The choices here are to propagate it (so that the consumer receives in the same transaction) or not. If the transaction is not propagated then there is a choice of whether to start a separate transaction for the consumer or not.
  c. Forward Dependency, this can be None, Commit or Abort. Commit means that the Consumer's transaction can only commit if the Producer's transaction commits, and Abort means only if the Producer's transaction aborts.
  d. Backward Dependency, this constrains when the producer's transaction can commit based on the consumer state. It can be None, Vital, or Mark-rollback
  e. Production of the Event could be included in the Producer's transaction or not
  f. Consumption of the Event could be included in the Consumer's transaction or not.

  These have been implemented by Stefan Tai of IBM Research in the X2TS middleware service.

- Scopes

  Publish/subscribe systems sometimes need to provide scoping arrangements such that certain consumers can only receive events from certain other producers. An example would be a wireless sensor network that is monitoring cargo containers. Each container might have many sensors reporting conditions in the container, but you don't want everyone to be able to receive that information. Scopes need to be hierarchical in nature, and each scope contains a set of producer and consumer nodes. The scope defines visibility of events and the ability to run certain tasks. Alex described an experimental system where each node was given a set of static and dynamic properties

which determined its scope - the dynamic properties allowed membership changes to occur.

2. **Oracle's Event Technologies (Dieter Gawlick); Dieter** gave a review of the various technologies that relate to Event Processing in the Oracle database

a. The first of these is the database Trigger mechanism. This has strange transactional behavior and wasn't designed for EP. Dieter's recommendation was not to use database Triggers for EP.
b. Messaging. The Oracle database provides a messaging system that is like JMS but with extensions. It is suitable for business messages, but not for everything - for example it shouldn't be used for high volume messages like stock ticks. The extensions beyond JMS include multi-consumer queues, delayed delivery and SQL access. The advantage of using Messaging in the database is that you avoid 2pc when combining a messaging operation with a database update, and you inherit the database's reliability, security and management models. SMP exploitation allows it to scale to 4000 messages/second. There were some technical challenges implementing this messaging such as index balancing and timing issues with multi-processors. Also the database needed to implement Skip Lock.
c. Rules. The messaging system allows subscriptions that include SQL rules selectors, and they also have an explicit PL/SQL-based subscription interface that applications can use directly. SQL only allows structures to be defined that are contained within tables in the database, and to support event structures that don't appear as rows in a table required 6 month's re-engineering. They now support Expressions as a data type and you can express subscriptions as rows in a table. This was highly scalable - a table can contain a million such subscription rules. These Expressions can refer to Data within the Event Message or data within other tables, and they allow subscriptions that are much more powerful than regular pub/sub, e.g. "Select consumer within a 5 mile radius". Dieter mentioned some other things that you could do with Oracle messaging that you can't do in standard pub/sub. These included ordering of subscribers and the ability to define Mutual Filters.
d. Rules Manager. Expressions in the database can be used to do ECA-style Complex Event Processing. There are expressions for Conjunction, Disjunction, Sequencing, and Non-Occurrence. These rules can be grouped into hierarchies and can be nested and re-used.
e. Flashback. This is a feature of the database which has been there for twenty years, but which has uses for EP. Each time an update is made, or a timer tick occurs the database state is logically saved (you can choose what is actually archived). A Flashback Query allows a query to be executed as if it were at some time in the past, a Flashback Version Query allows you to see all versions of a row between two specified times, and a Flashback Transaction Query allows you to see the changes made by a transaction.
f. Continuous Query Notification. Users and applications can register Queries in the database in a completely declarative manner, and receive in-memory notifications. These are transactional but not persistent. However they can be used against historical data using the Flashback capability. This allows you to try out a new Query against historical data before deploying it against live input streams.

The fourth Day: Thursday

Session 5 - Positioning of SOA/EDA/BPM/CEP

1. **David Luckham; David** stated that SOA/EDA/BPM/CEP are all complementary. Here is a rough transcript of his words:
   "What we are talking about here are Design Philosophies. Design Philosophies are important because they persuade people in business that hold the purse-strings that it is time to dip into their pockets again. These people have heard many promises over the last 20 years and each time they have been disappointed. Now we need to persuade them to dip into their pockets one more time, for Event Processing. Our story has been complicated, fuzzed up one might say, by all the buzzwords that are bandied about in the industry. For example there is much argument at conferences about the definition of SOA - even Wikipedia admits that there is no agreement about what SOA is. So let us start with the definition of a Service as a Function with two top-level concepts: *Modularity* as defined in Computer Science and *Remote Access*. Modularity means that Services have Interfaces and Metadata, and they can be logically grouped together to form Modules. The separation of implementation from interface means that, in theory, a user of the service does not need to know details of its implementation; indeed the implementation can be changed without affecting the users."

   "Remote Access has typically meant access by Request/Reply message exchange patterns (also referred to as Remote Procedure Call). This is the 2001 version of SOA, based on CORBA. We all know that this is inefficient and waiting for replies leads to long delays. A better approach is to use Event-Driven remote access, where all interactions are via messages - let us call this ED-SOA (Event-Driven SOA). The downside of ED-SOA is that it opens up a lot more opportunities for crooks, and the fact that it enables much higher volumes of events brings scalability challenges for implementers."

   "You can think of SOA as a design methodology for EDA applications, or putting it another way an EDA is a SOA in which all services are reactive event-driven processes and all communications between services is via events. SOA can be applied to Business Processes to make them easier to build, and Business Process Monitoring (BPM) systems are examples of ED-SOAs, where CEP is used to monitor the Business Processes. But you can go further and collapse the monitoring and feedback bits of BPM into the Business Processes themselves to make autonomous Business Processes."

Alex Buchmann asked about the role of data. Does data flow with events, or is it another service?

2. **A perspective from the field (Marc Peters); Marc**, an IBM software IT architect, gave his impressions on how EDA is being viewed by the Energy company customers he works with. Events are used to start services or processes. Services are used to

capture, react to, or process events, and can themselves generate Events. Events are often stored and also are used to trigger human activity. At present many events are confined within closed - often proprietary - systems e.g. SCADA systems or IT event monitoring systems.
Customers today have a mixture of different infrastructures, some use message oriented middleware, some have started with Service Orientation, and some haven't. Most have Business Intelligence systems. The only common factor is that they all store data. Going forward they are looking to gain flexibility by using SOA and applying BI principles to streamed data as well as stored data. They are also interested in integrating events from sensor networks with their business systems.
Comparing SOA and EDA, **Marc** thought that both were architectural styles which handled separation of concerns and lifecycle/governance, and which could be implemented separately or together. EDA is often referred to as decoupled, as opposed to SOA which is loosely coupled - but things aren't as simple as that.
Marc concluded that the value of EDA, BAM, SOA could bring value to customers, but the value had to be proven before we would see widespread adoption. EDA needs an Entry Point - BAM is the most promising but RTE (Real Time Enterprise) is also a possibility.
Dieter Gawlick commented that we need to consider data and its consistency; otherwise this is a house of cards.

3. Roy Schulte; Roy started by saying that, to the analyst, SOA was Business Process Re-engineering 2.0. The key issue was aligning the IT design with the business architecture. In SOA we replace monolithic applications with multiple services owned by different autonomous business units. However the definition of SOA depends on who you talk to. It has been applied to

- Any distributed application
- Any system with well-defined interfaces
- Anything with WSDL

Roy suggested 5 principles for SOA

i) Modular design
ii) Modules can be distributed onto different servers
iii) Interfaces that specify explicit contracts, a third party can find out about a service by examining its contract
iv) Separation of interface from implementation
v) Services are serially reusable

Events can appear in several contexts. They can be passed as Messages in their own right, they can be passed as arguments in a RPC-like Request/Reply message exchange, or they can be stored in a file or database (Events at Rest). Roy suggested that we use the term EP to refer to all processing connected with Events, and use the term EDA to refer to Events passed as Messages. In traditional Request/Reply the interaction is initiated by the Producer and the Service responding to the request is known to the producer; in EDA neither of these is true, the interaction is initiated by the Consumer, and the Consumer is unknown to the Producer.
Roy also suggested that we should use the terms SEP (Simple Event Processing) and CEP to refer to the processing applied to Events, regardless of whether they are passed as Messages, passed as RPC parameters, or stored in databases.

4. Shared Model - Mani Chandy; Mani started by saying that one of the objectives of Dagstuhl was to achieve coherence, and interestingly the idea of achieving

coherence is important in event-driven communications. If two people have been married for a long time they develop a common shared model of the world, which forms an implicit contract between them and they only need to communicate if something untoward happens that affects that model. In the same way if you take a network of intelligent servers you can consider them as all sharing a common model and each server only needs notify the others if it detects something that deviates from that model.

Mani applied this to **Katherina Hahn'**s avalanche prediction scenario. If the base station and the sensors were to have a shared model for what the air pressure should be, then it would not be necessary for each sensor to notify the base station every time it detected a change in pressure, it would only need to notify it if it detected a change that deviated from this model. Similarly in the healthcare scenario; where the problem is information overload and a flood of false positives, the system needs to make sure it only communicates events to the end-user that deviate from that user's model. In this scenario you can observe a progressive filtering of events as you go from the sensors themselves, to the monitoring equipment, to the nursing staff, and then to a succession of more highly paid doctors. At each stage the receiver of events has a more sophisticated model and a lower tolerance of false positives. So you would expect a large number of false positives at the early stages of the filtering process, and a much reduced number at the final stage when the most senior doctor is called in. Security of the shared model is also important, particularly in military applications. We often think about protecting the events and information but it is also important to protect the model.

He then turned to the discussion of SOA and EDA, and asked what is different now. He thought the many difference is that Enterprises are now looking outside. From 1960 to 2000 the focus of ERP systems was on managing things inside the enterprise where things were more or less under control. Data was reasonably reliable and the enterprise could choose the schemas used to represent it. Now enterprises are becoming more reliant on data supplied by partners - for example an energy trader needs data from its suppliers. This means the data is going to be less reliable so probabilistic approaches like those discussed by **Avi Gal** are becoming more and more important. The probability issue is related to the partial data question discussed by **Jonathan Goldstein**. Under what circumstances should you notify someone of a compound event if you have only got partial input data? This depends on the tolerance of the consumers towards false positives, and their ability to handle later retractions. This brings us back to the idea of producers and consumers sharing a common model. We can apply the shared model idea to the SOA/EDA question. In a Request/Reply paradigm the caller has complete control. It submits a request and gets a well-defined, deterministic result. This is easy to understand, explain and teach to people who are learning to program. However it is not the paradigm used in Multiplayer Online Games, which instead use the shared model approach (avatars only signal events that affect the shared model, just as a frog's eye doesn't return everything it sees to the frog's brain) - and as we can expect more new programmers to be familiar with such games it should become easier to explain the shared model approach. You can view BAM as using a shared model – it's the set of KPIs that constitutes the model.

Mani related the idea of the shared model to the idea of Topics as discussed by **Peter Niblett**. You can view a Topic as a set of models, and by subscribing to a Topic you indicate that you understand those models and therefore wish to receive events that relate to it. These could either be events that deviated from/change the model, or "continuation events". He thought that the WS-Topics specification presented by

Peter had some problems, as it didn't enforce a relationship between a Topic and a SubTopic. It also tied Topics to Event Schemas.
There has been a lot of theory on data models, and we now need to move forward to Reality models. **Vijay Dialani** had talked about Statistical models, Reality Models and BI models, so we can see there could be a stepwise refinement of models with a separation of concerns between these models. Given the success of data models it is possible that adding time into the model might give a useful Reality model, when taken with some of the capabilities discussed by **Dieter Gawlick** such as Flashback.
Mani concluded by saying that several of the talks had shown there is a confluence of Event-Condition-Action with sensor networks and messaging. The idea of a shared model is central to this convergence as it is all about how you understand reality.

Continuation of Session 4 – Implementation

1. **Sensor networks and Grid-based EP (Eui-Nam Huh); Eui-Nam** described a project involving wireless sensors (connected to the human body), realtime processing and Web 2.0. Wireless sensor networks (using 802.15.4) are fairly mature, but suffer from errors and lost data. What they needed was integrated technology for sharing real-time events efficiently. It has to be capable of handling large volumes of events and provide functionality such as storage, transformation, aggregation and derived events (periodic or a-periodic). They had looked at Grid standards, and selected OGSA-DAI (an extensible framework for data access and integration) and the INFO-D dissemination service, with the EAQ event aggregator. The hardest thing to implement was the matching between publishers and subscribers, and they had to develop their own algorithms for this. They used a Class Group Matching approach, where each subscription were described using ranges, and then the matching process was optimized by grouping subscribers together. They did a three pass match, the first pass being a course one, the second a dynamic fine-level match and the third step being an exact match. Doing the process this way saved 2/3 of the matching time, when compared with a sequential match process.

2. **Denial of Information Attacks in Event Processing (Calton Pu); Calton** said that his aim in this talk was to make us all feel insecure. Spam and Denial of Information is a real problem for email, and we can't assume that Event Processing will be immune from the same kinds of attack. He started by quantifying the email problem. In the first quarter of 2006, the Messaging Anti-Abuse Working Group analyzed 390 Million mailboxes and found that they had blocked 370 billion emails out of total of 460 billion - i.e. 80% of all emails. A fair number of the 20% that get through are still Spam, In Calton's case he reckoned that was another 80% so that meant that only 4% of the mail addressed to him is legitimate. The problem is that it is easy and cheap to generate Spam; Moore's law helps the spammers. The problem is not restricted to email. There's also Web Spam (20% of static pages are spam), Blog Spam and Spit (Spam Voice over IP). He then gave a brief history of the arms race between Spammers and the Spam filters - simple keyword filtering had been circumvented by spelling mistakes, blacklists circumvented by spoofed headers, use of image to defeat text-based anti-spam filters, use of wavy text in images to defeat OCR anti-spammers. One of the problems is that Spammers are able to register their own accounts with mail providers and so they can spam themselves - continually tuning their spam

until it gets through. The time lag between a spammer finding a way through a filter and that way through being blocked off is sufficiently large that spamming is profitable.
There is no end in sight to this arms race, but some spam filter approaches can be quite successful. The most promising is to look for emails which contain deliberate misspellings of keywords (such misspellings would never appear in legitimate messages), and simply block these regardless of how much innocuous camouflage the email contains.
In answer to questions, Calton said it was thought that were only about 100 spammers in the world, mostly in Florida, but it was very hard to prove that someone was a spammer so catching them wasn't a viable option. There was a discussion of whether there was a way to remove the financial incentive.

3. **Content-based subscriptions in distributed networks (Antonio Carzaniga); Antonio** discussed approaches to implement content-based publish/subscribe using a network of brokers. The quality of service to be provided is "best-efforts". Each consumer provides a filter expression and the idea is to distribute the filtering among the brokers to achieve the desired result in an efficient fashion. The challenges faced in the general case are
    1. Matching between producers and consumers when they use different terms in their filter to refer to the same content (e.g. Soccer and Football)
    2. If the network consists of multiple autonomous regions, that the owner of each region is unlikely to allow internal routing information to be disclosed to other regions
    3. Complexity of the content-based selection
    4. Choice of routing protocol
    5. Consumer privacy. A consumer's selection filter might reveal sensitive information about that consumer. The owner of the filter might not want it propagated to other servers.
    6. Criteria for evaluating the efficiency of the system

    Antonio showed results of a couple of routing schemes. He evaluated them by looking at the time to achieve stability if there was a change of subscribers, amount of false positives, e.g. messages delivered to consumers that did not match their filters, and memory consumption at each node. Another criterion would be time taken to do the match. In particular he was concerned about the memory usage as the number of producers, consumers and network nodes increased.

4. **High Performance BPM (Lingzhao Zeng); Zeng** talked about techniques to improve BPM performance - in BPM we take events and use them to update metrics, we then analyze the metrics to detect Situations. The individual event filters are quite simple, but there can be a large number of them and there is also the requirement to persist data. He examined three techniques
    a) Model driven transformation - The idea here is to refactor the observation model at deployment time in order to optimize runtime access. You can rearrange the data to make the access more efficient. You can also generate custom Java code and pre-compile it rather than using general purpose evaluation engines. At the same time you can generate the SQL statements needed to persist the data, and generate an in-memory materialized view.

b) Cluster exploitation - Distribute the queries across a network of processors. The difficulty here is partitioning the work so that rules that update the same metric don't execute concurrently. It is also hard to maximize CPU usage at each node.
c) Execution optimization - Henk de Man asked for more details, Zeng indicated that he would be publishing a paper in the IBM Systems Journal.

5. **Twelve theses for Reactive Rules (Francois Bry); Francois Bry** is a member of the W3C Rule Interchange Format working group (part of the W3C semantic web activity). He started by saying that a lot of people had been dreaming of a universal rules language. The idea of this was that you could translate into this universal language from any existing language without loss of information. He was dubious about whether this was possible, but said that reactivity was important, especially in adaptive Web services, and so instead focused on characteristics that such languages should possess (we are talking here about high level languages which are abstracted away from any communications or implementation details). He presented 12 theses which are available in his presentation at http://kathrin.dagstuhl.de/files/Materials/07/07191/07191.BryFrancois.Slides.pdf **Opher** commented that we had fixed on ECA back in 1994, but it was now time to move on. **Dieter Gawlick** said that CEA was important as well. There was then a discussion about whether there was a real difference between Event and Condition.

6. **Event Processing and Publish/Subscribe (Arno Jacobsen); Arno's** main thesis was that Event Processing should be based around content-based publish/subscribe. This could be implemented either as a centralized broker or as a network of distributed pub/sub agents - the choice of implementation topology was immaterial to the pub/sub abstraction. He gave an example based on shopping at a website like Amazon. During the course of your normal shopping the website builds a profile about your shopping habits. When you visit a new city the website could push you a message telling you about books in stock at a local bookshop. He classified publish/subscribe systems into four types: Channel-based, Topic-based, Type-based, Content-based, and State-based. There's now a variety of hardware implementations of publish/subscribe available. Arno is researching into expressive subscription languages, matching algorithms and routing protocols. He also has a project called PADRES which is looking at using publish/subscribe in connection with Business Processes. The idea is to take business processes expressed in BPEL and execute them over a publish/subscribe infrastructure (it needs a database as well of course). The same infrastructure can be used both for the Business Process itself and for any Monitoring of that process. The main challenge is ensuring that it is robust. The administrator sets up an overlay network of pub/sub brokers and the BPEL engine sends control messages to these brokers to establish subscriptions - it translates dependencies between business processes into subscriptions. This allows the processes to run concurrently on different systems.

7. **Parallel processing in Scientific EP (Tore Risch); Tore** discussed some of the implementation challenges arising from the LOFAR telescope project. The basic approach is to define Continuous Queries running over windows in the streamed

data; these queries are continuous in that they continue to run until explicitly stopped or until they detect a specific stop condition. The incoming data is time stamped and the query functions are typically computing aggregates over the data. There are several strategies for parallelizing this in a shared-nothing multi-processor network, depending on the type of query

i) Window Distribution. Allocate different windows to different processors on a round-robin basis
ii) Window Split. Distribute a single (large) window across multiple processors. This works for some kinds of aggregation (e.g. FFT)
iii) PCC, a combination of the other two patterns where the stream processing is split into separate steps and different partitioning can be used at each step.

When deploying this to the IBM BlueGene with 100,000 processors the problem becomes very dynamic. Finding processors becomes a database problem. The Windows Distribution approach seems to work OK regardless of the number of processors.

The processing has to be reactive, in that the processing required can vary depending on the results being returned by the processing.

8. **Event Processing in wireless sensor networks (Kirsten Terfloth); Kirsten** described the implementation of the Fence monitoring example. In doing this implementation they had tried to achieve
    a) Suitable abstraction from the particular hardware chosen
    b) A modular design that could be reused in other application areas
    c) A resource-aware implementation

    She had designed some rule-based middleware called FACTS which provided an event-centric language, based on named tuples (Facts) and algorithms (Rules). There is a repository of Facts and Rules are triggered when a new Fact enters the repository.

    The processing in each sensor is split into several steps. The first step is to filter the raw data via thresholds to generate raw events when a threshold is crossed. In the second step they look at the number of raw events generated in a given time (this step uses the FACTS language) and generates a new event if a sufficient number of raw events were detected. In the third step the sensor node asks its neighbors if they can confirm this observation. If they can then the sensor reports the event back to the base station. It does this via multi-hop through the other sensors.

    The objective of the exercise was to validate the FACTS language, rather than produce the ultimate physical intrusion detection system. The implementation of the rules engine used 8kb of code in the sensors

The fifth Day: Friday

Final Session:

Recommended Research directions:

The organizers will provide a white paper about the research directions of what agreed to be the most important topics to promote the state of the art in event processing, with more details about each of them;
Table 1 provides short description of the various in titles only.

Table 1:

Num	Topic	**Short Explanation**
1.	Event Processing Algebra and meta-language	Unify all concepts from different industrial implementation in a single algebra that will provide a unified formal basis, and be a source of a common meta-language that will be taken to standard: the different vendors will be able to support it using various implementations
2	Software Engineering and Modeling issues	Software engineering models and practices around event processing will have crucial role in the practical success of the EP disciplines; research topics include – model-driven approach to EP logic specification, modeling semantics in UML, design methodology, debug and validation tools, non-technical people enablement (model and visualization)
3	Implementation issues	Optimization for various cases, parallel processing, handling of high throughput, support in real-time constraints, security issues, transactionality and transaction models, inter-operability issues, hardware acceleration
4	Pragmatic issues	Involve academic people in common terminology; getting use cases from industry (functional and non-functional requirements) to academia, to help tune up the research, teaching event processing courses, awareness cross core research disciplines, clear positioning with

		respect to "business rules".
5	Various issues	Event retention and vacuuming, handling of out-of-order events, handling of uncertain event (various types of uncertainty).

Short term action items:

1. Publish seminar material:
2. Advertising more EP oriented conferences/workshops:
    a. DEBS - http://debs.msrg.toronto.edu/index.shtml Toronto, June 20-22.
    b. ICDCS workshop (DEPSA) - http://www.cs.uga.edu/~laks/depsa/index.htm Toronto, June 29
    c. Third EPTS event processing symposium – tentative – Orlando, September 17-19.
    d. Gartner Event Processing summit – http://www.gartner.com/it/page.jsp?id=502259&tab=overview, Orlando, September 19-21
    e. VLDB workshop (EDAPS) - http://www.cc.gatech.edu/projects/disl/conferences/EDAPS/ Vienna, September 24. Call for papers is still valid: May 18 – deadline for abstracts; May 25 – deadline for Paper. Your next opportunity to submit your work.
3. Publishing the seminar summary in ACM SIGMOD RECORD – get awareness in the database community. The organizers are responsible – call for volunteers to help will be issued
4. Participation in EPTS workgroups - calls will be issued.
5. Establishing Network of Excellence – EU: Alex Buchmann, Francois Bry, Avi Gal – will make the initial approach and notify all interested parties.
6. Encyclopedia of Database Systems – solicit volunteers for event processing terms. A call will be issued.
7. ACM SIG – there has been a long discussion for and against trying to establish an ACM SIG in the current point. A (small) majority decided to proceed with the application for ACM SIG. A call for support letters will be issued.

Long Term Actions – some of the initial actions are short term:

1. Book of articles: A book that contains important articles in the state of the art (both new and old, with copyright permission) has been proposed. A call for editors will be issued short-term to jump-start this activity.
2. Teaching event processing: some courses already exist, typically in the level of advanced/elective courses. A textbook is needed, but probably premature, until we'll advance in the common algebra/meta-language. Meanwhile, a teaching portal with collection of teaching materials that already exists will be constructed. A call for material will be issued.
3. Federated Conference: There has been a discussion about the need to have a single annual conference, which may be a federation of several existing

conferences and workshops, but not associated with any major core-discipline conference. Discussion among organizers of this year's conferences and workshops will be set.

4. Special issue of journals: Two proposals – a popular magazine (e.g. ACM CACM or IEEE Computer) and scientific journal --- ACM/IEEE Transaction… -- call for volunteers to pursue these journals will be issued.
5. Summer school: There was a proposal to do summer school on EP, an EU network of excellence may be a way to fund such a summer school, thus this item is tabled until further progress in the network of excellence is achieved.
6. Event processing journals: newsletter and transactions type journal – will be proposed as part of the ACM SIG proposal.