

- ❖ what do we mean by event processing... **now**,
- ❖ complex event processing,
- ❖ a checklist of capabilities in current event processing tools and applications,
- ❖ next steps in developing event processing as a science.

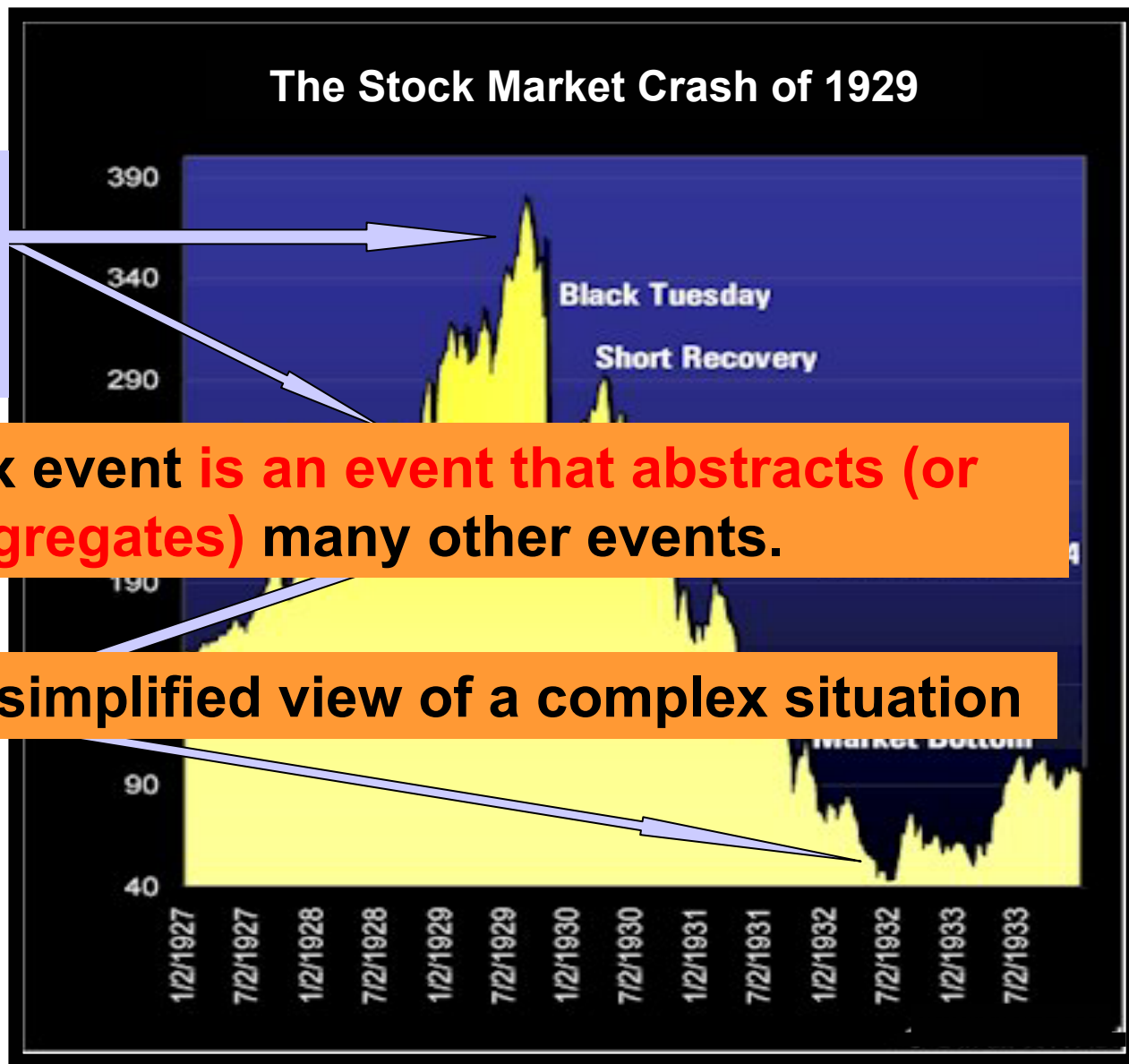
What do we mean by “Event Processing”

- ❖ **and ultimately,** the fully autonomous event driven enterprise. Event Processing has meant different things at different times.
- ❖ **1990’s – 2000’s:** EAI, SOA, BAM, BPM, EDA, and ... **CEP**
- ❖ **1980’s – 90’s:** ascent of **Middleware, Integration and Services**
- ❖ **1980’s:** an explosion of **client-server computing, stove pipe systems, chaos and legacy ...**
- ❖ **1960’s – 1970’s:** **desktop applications, email, Data Bases, event driven simulation, ...**
- ❖ **1960’s:** **operating systems, networks, packets, network protocols, ...**

SECURITY

- ❖ **Hierarchies** of events are a fact of life in the 2000's. A result of the archeology of IT layers (actual and virtual) that has grown up over the past 50 years.
- ❖ Some of today's applications focus on a single layer or a few types of event.
This is motivated by the kinds of commercial problems that are currently in fashion.
- ❖ Complex event processing defines a set of event processing techniques that includes complex event patterns and hierarchical abstraction.

The 1929 Stock Market Crash

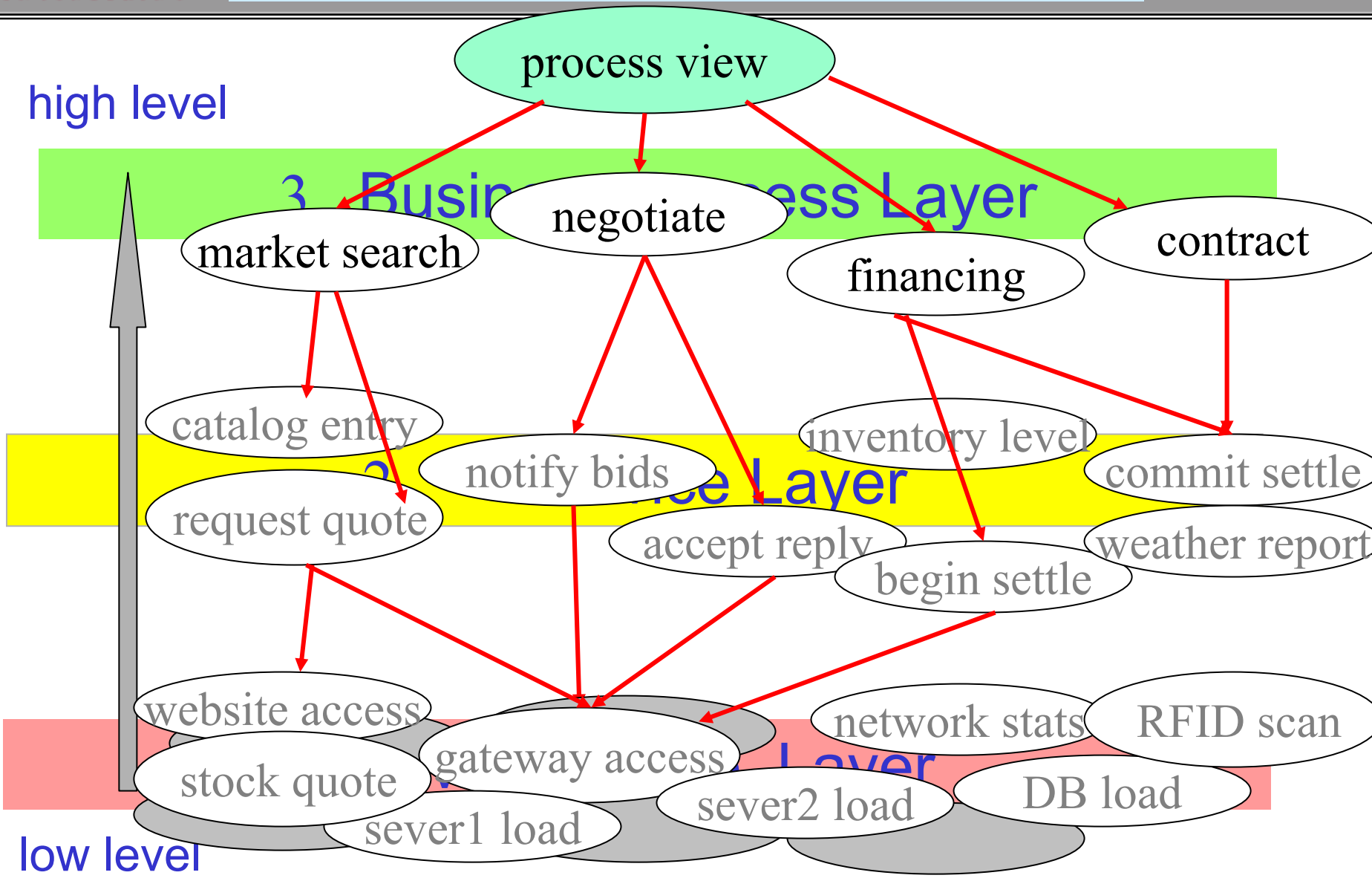


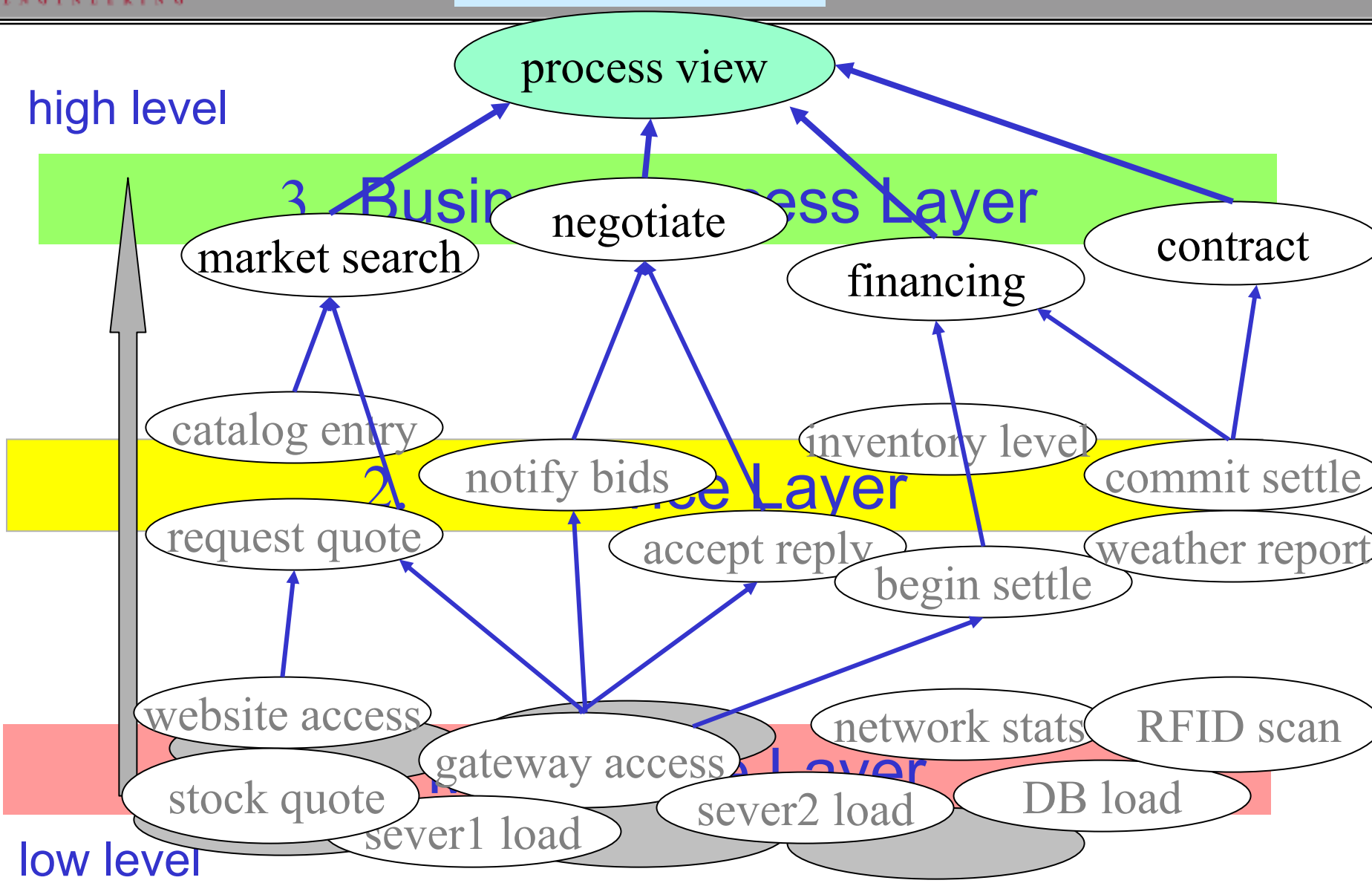
events resulting from financial euphoria leading up to the ...

A complex event is an event that abstracts (or aggregates) many other events.

It presents a simplified view of a complex situation comprising the downward spiral

dependence between events





Enables insight needed for business decisions at highest level

high level

3. Business Process Layer

Aggregating event patterns into higher level events

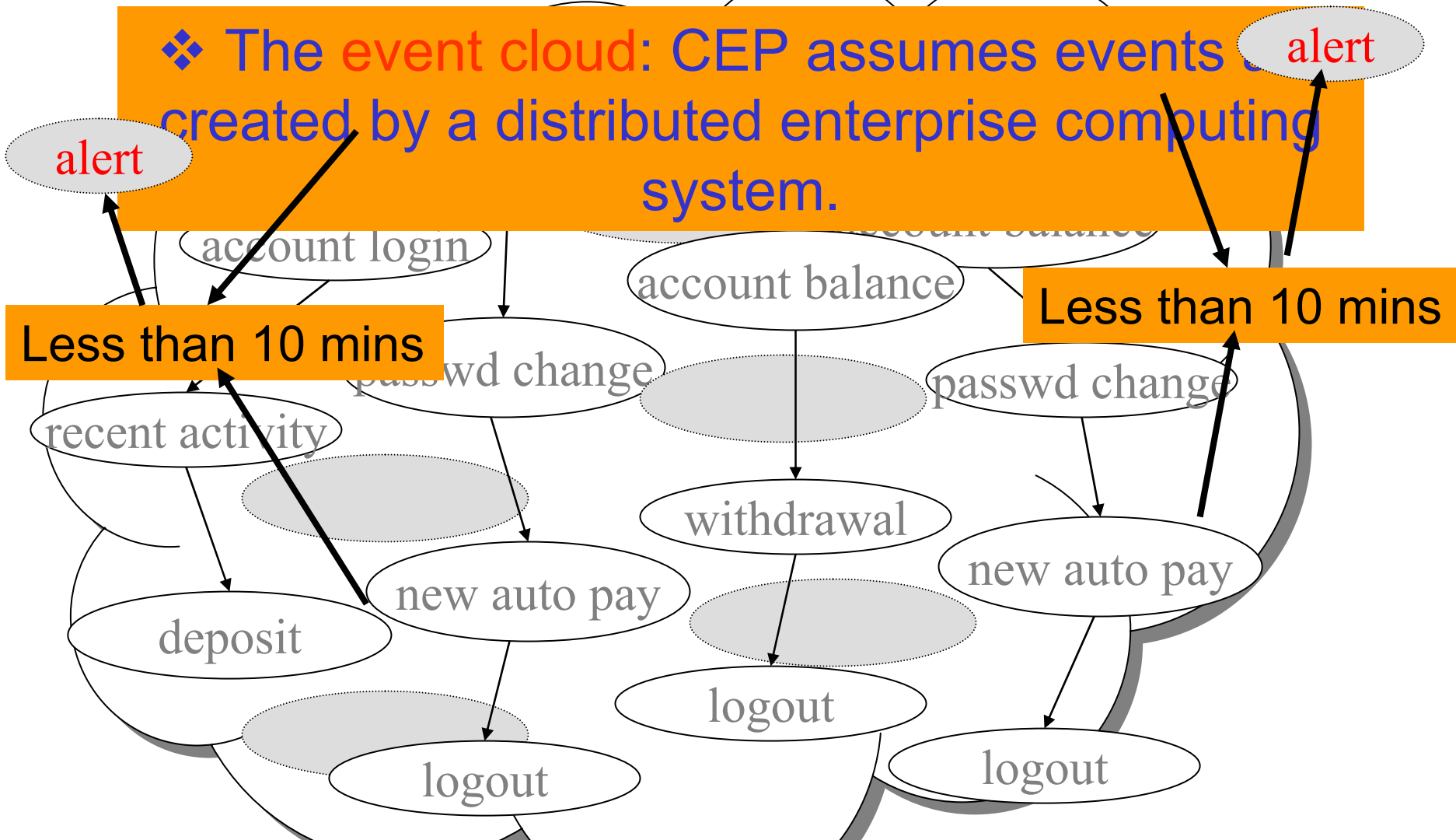
2. Service Layer

Detecting patterns of events at all IT levels

1. Event driven middleware Layer

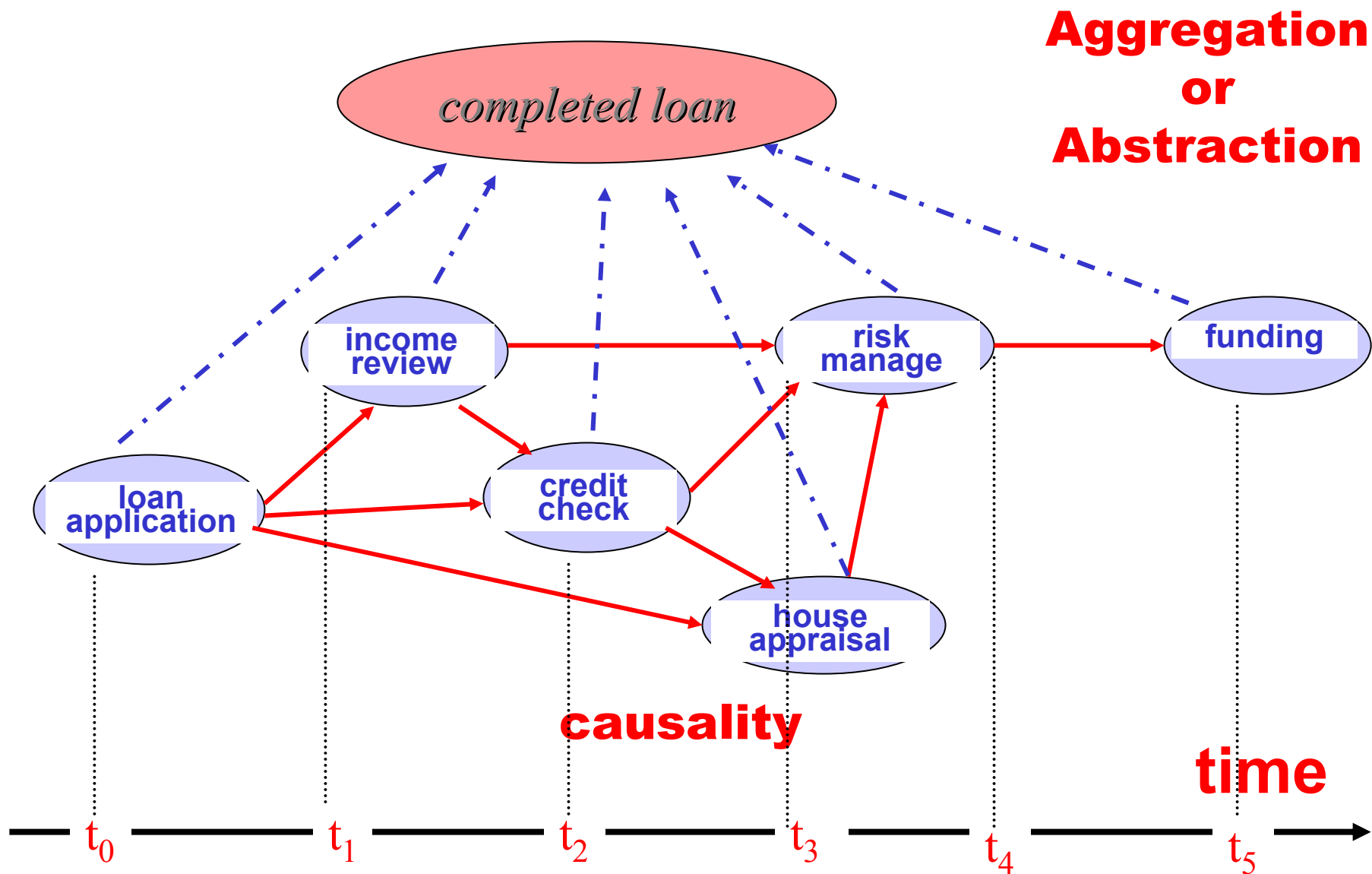
low level

❖ The event cloud: CEP assumes events created by a distributed enterprise computing system.

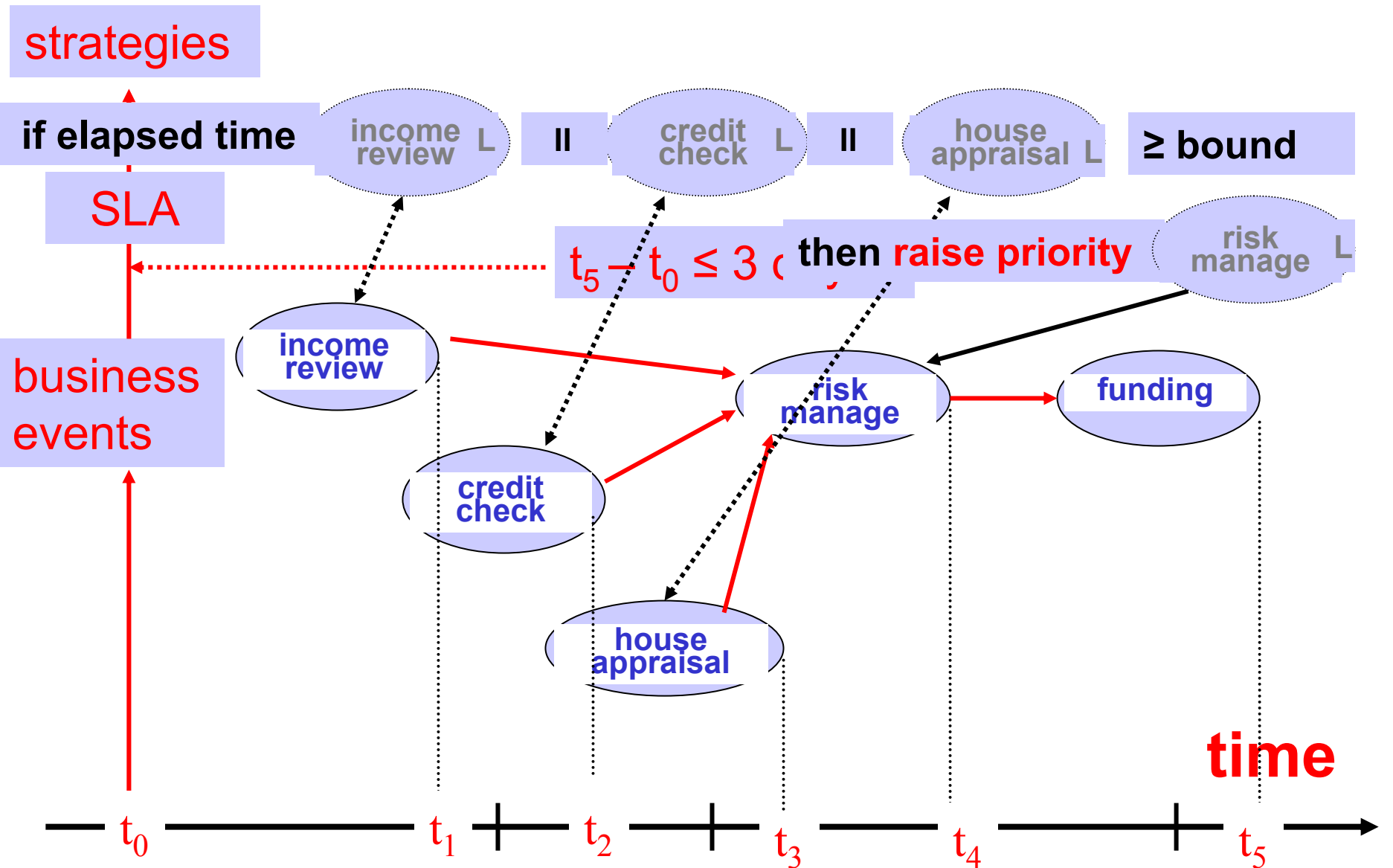


The event cloud in an on-line banking website

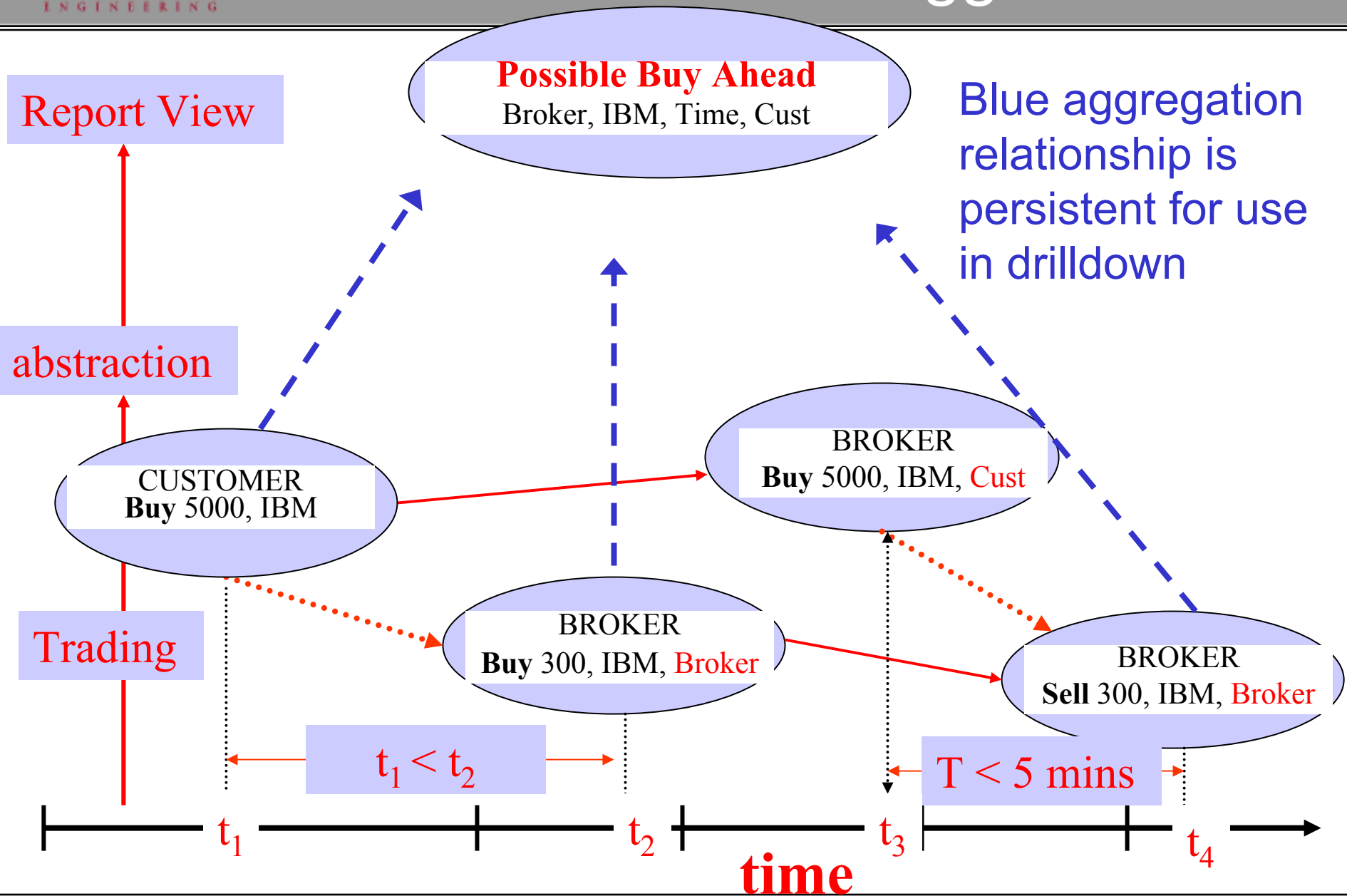
Complex Event Patterns Involve Events & Event Relationships



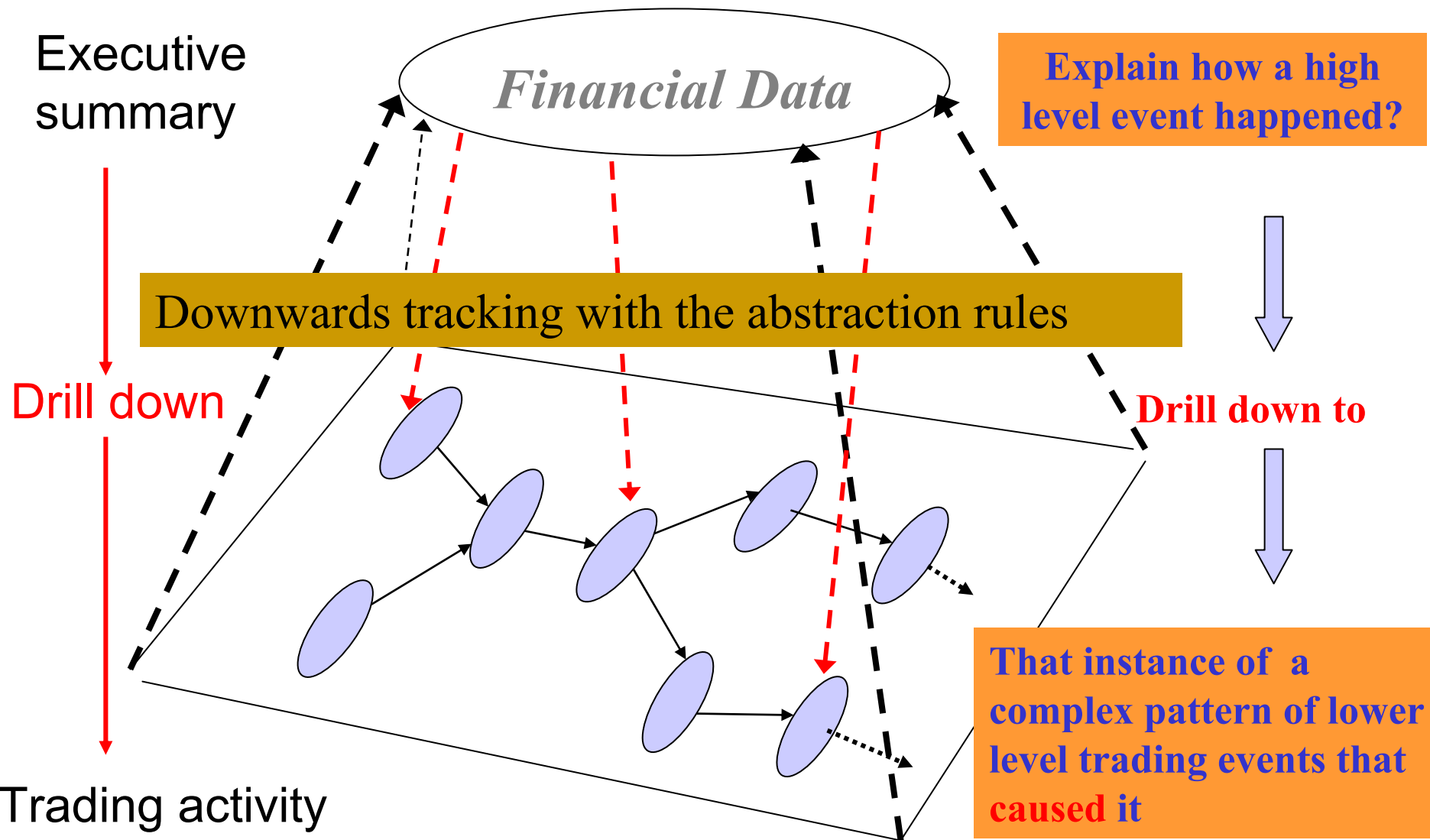
Event Patterns to Control Process Activity



Event Pattern Triggered Rules



You want an explanation?



You want an explanation?

Executive
summary

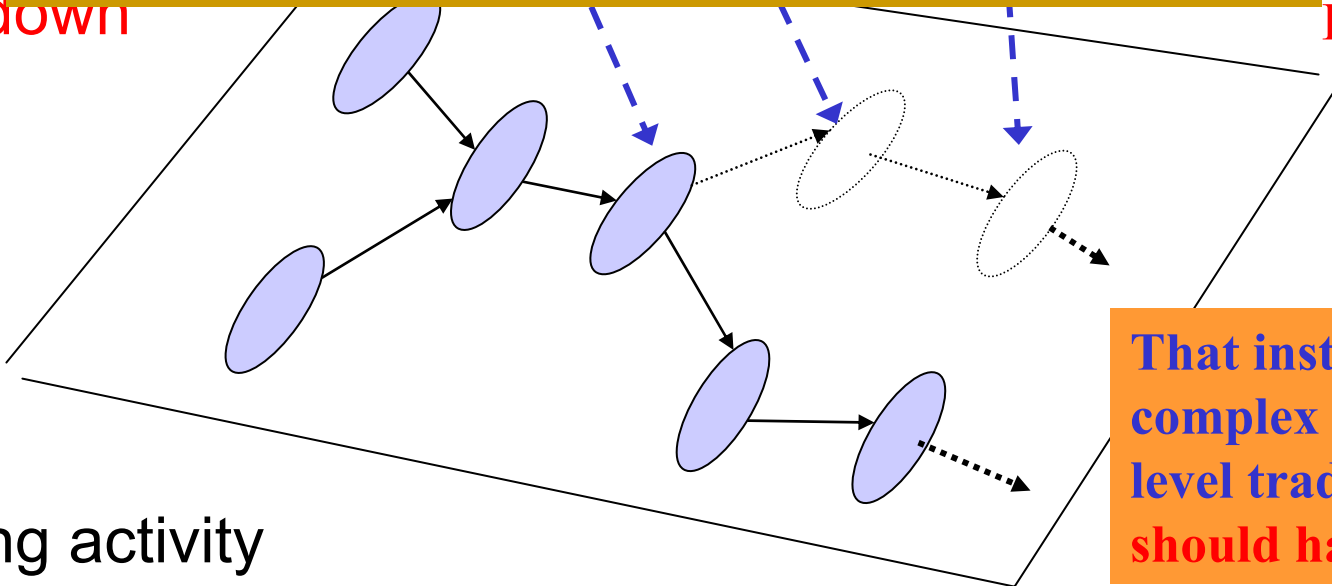
Didn't happen!

**Explain why a high
level event didn't
happen?**

Abstraction rules didn't trigger because of
missing lower level events

Drill down

Drill down to



**That instance of a
complex pattern of lower
level trading events that
should have caused it**

Trading activity

Certain kinds of event processing applications don't use patterns of events at all.

They simply use the data contained in the events.

Other kinds of applications can use some or all of:

- (1) complex event patterns,
- (2) the data in events,
- (3) state, and
- (4) hierarchical abstraction.

A Checklist of Event Processing Capabilities

An event processing tool or application may possess one or more event processing capabilities. Here is my personal checklist:

- 1. Real-time computation on data in events.**
- 2. Simple event pattern detection and reaction.**
- 3. Event streams processing.**
- 4. Detection of, and reaction to complex patterns of events.**
- 5. Event pattern abstraction.**

The application can compute on the data contained in events in real time.

- Typically,
- to search for sets of events containing unusual data,
 - events containing related data,
 - statistical analysis of data in large sets of events

- Examples:**
- Some types of conformance tools (e.g., money laundering detection), (**often COB time**)
 - air line seat pricing (**COB analysis**),
 - Data Streams Management tools. (**“Right Now” time**)

The application can detect (instances of) single events, or Boolean combinations of events, in conjunction with state values, and then take actions.

- ❖ events to be detected are either predefined or defined by the user. E.g., events indicating critical situations or KPIs.
- ❖ pattern detection triggers reactive rules to take actions. Rules are also predefined or specified by the user. E.g., Alerts, change process priorities.
- ❖ Actions may involve sending email messages, performing computations, etc.

Examples: Most of the current generation of Business Activity Monitoring tools.

The application processes one or more input streams of events in real time.

- ❖ An **event stream** is a real-time, continuous, ordered (by arrival time at the tool, or by a time stamping mechanism) sequence of events ("Data Stream Management Issues - A Survey", 2003).

- ❖ The application processes event data in order of arrival.

Most ESP tools have strategies for out-of-order arrival of events. How performance is affected is to be learnt here. **optimized event throughput.**

Examples: processing stockmarket feeds to compute metrics such as VWAP, deviations from averages, trade volume surges, etc. Typically used in algorithmic trading.

Focus is on high event volumes. 100,000 events/sec are currently being claimed.

The application can detect complex patterns of many events in the global event cloud of the enterprise and react to them according to defined rules.

- ❖ A complex pattern is made up of many events, often created at different locations in an enterprise and possibly in different time zones. Detection may involve long time periods.
- ❖ A complex pattern may involve creation time, causality and independence of events, and state.
- ❖ There is no assumption of correct order of arrival.

Examples: ● monitoring, analysis and optimization of concurrent and collaborating business processes,

- detecting patterns of fraudulent activity,
- cyber defense,
- conformance to business policies or regulations.

Problem: Monitor for lack of coordination of business processes operating in NYC and Tokyo.

detect a pattern of four events, A,B,C and D, where A happens in New York independently of event B that happens in Tokyo (i.e., A and B are causally unrelated), and event C is a DB update happening in SF caused by A, and event D is another update of the same DB caused by B.

**Event X, Y; $X = A(\text{origin is NYC}) \ \& \ Y = B(\text{origin is Tokyo})$
 $\& \ X \parallel Y \ \& \ X \rightarrow C(\text{origin is SF}) \ \& \ Y \rightarrow D(\text{origin is SF})$;**

The main point is to express the two relational operators \parallel (meaning “**happens independently**”) and \rightarrow (meaning “**causes**”). Note that “ \rightarrow ” is not the same as order of arrival. I have omitted placeholders for data parameters of the events.

The application creates new events whenever a complex event pattern matches. The new events abstract information from the event pattern.

- ❖ Important data is included in the new events, and unnecessary details are omitted. The new event is an *abstraction*.
- ❖ Event pattern abstraction is a method of supplying higher level views of complex patterns of business activity.

Examples: • Event streams processing can be viewed as supplying abstractions of **intervals of events** in streams.

- Event pattern abstraction is employed in the more ambitious of the current generation of BAM tools, but without definitions of reference hierarchies .

1. Higher level languages for event patterns and rules, including temporal, causal, and similarity event relations.
2. Off-the-shelf scalable engines for event pattern matching and rule execution using higher level languages.
3. Rules management tools for large sets of rules.
4. Libraries of event processing agents for building event processing applications.
5. Standards for event hierarchy definitions, and standard hierarchies.
6. Development of methods of specifying and designing event driven architectures.

1. Higher level languages for event patterns and rules, supporting a rich selection of event relations (e.g., **temporal**, **causal**, and **similarity** relations) need to be developed.

Current approaches include Javascript, FSMs, SQL, StreamingSQL, UML, ...

Are these formalisms sufficient for future event processing?

Can we sustain large scale event processing applications involving thousands of patterns and rules using Java code or FSM models? Will we lose control of our rule bases?

Can we develop effective rules management tools for rules expressed in the current formalisms?

2. Off-the-shelf scalable event pattern matching and rule execution engines supporting the higher level languages.

We are beginning to see the advent of both commercial off-the-shelf and open software event processing execution engines. We need more!

These engines need to be scalable and to support higher level languages for complex event patterns and rules. They will form a basis for building CEP applications.

Scalability involves tackling the problems of scheduling sets of cooperating EPAs, multiple engines, grid computing with these engines etc.

3. Rules management tools for event processing rules.

As event processing applications become more ambitious, involving thousands of rules, the management of the rules will become a pressing problem.

Management of patterns and rules will entail automatically answering questions such as:

- ❖ “if this pattern matches **then** that pattern will/won’t match.”
- ❖ Are these rules **inconsistent**, or **incorrect**,
- ❖ **mutually redundant**, and
- ❖ how to structure large rule sets to optimize execution.

4. Libraries of event processing agents for building event processing applications.

An EPA is an autonomous modular rule set with local state that takes events as input and produces events. An EPA is a generic module that can be instantiated and configured in an Event Processing Network of EPAs.

Off-the-shelf event processing agents will become the building blocks for event processing applications distributed across the enterprise IT infrastructure.

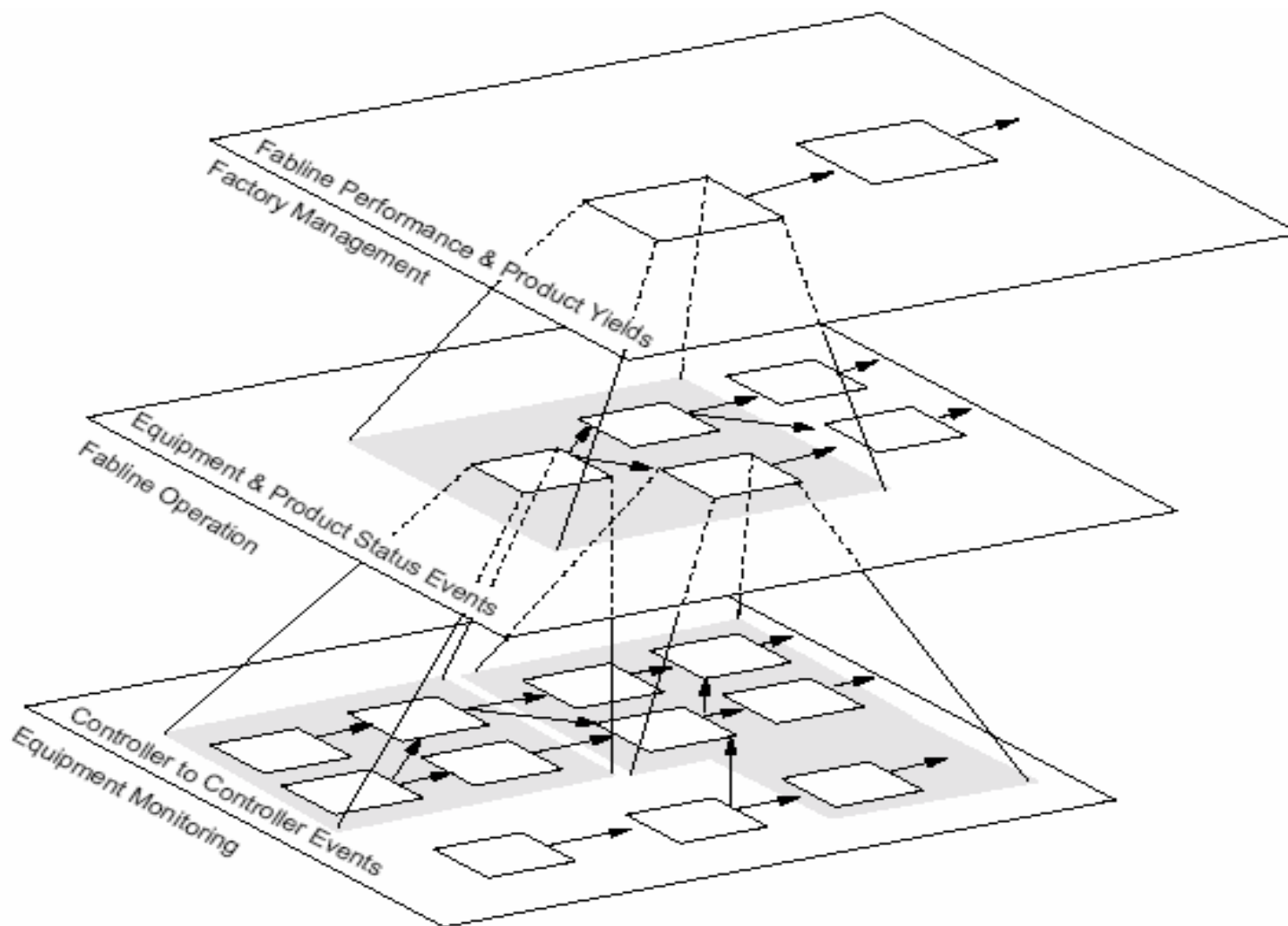
Standards for event hierarchy definitions, and standard hierarchies.

- ❖ What methods should be used to define event hierarchies for particular problem domains?
- ❖ What hierarchies are good candidates for standardization?

Examples: TCP/IP protocol hierarchy (1969),
ISO 7 layer messaging hierarchy (1983),
Standard chip design hierarchy (1970's).

Suggestions: -- Hierarchies of events in business processes based upon process models.
-- Hierarchies of RFID tracking events.

Example hierarchy in Manufacturing



Development of methods of specifying and designing event driven architectures.

Industry is making good progress toward defining notations and development methodologies for SOA.

Can EDA methodologies be “piggy backed” on the SOA efforts?

Should we try to develop a roadmap for how event-driven IT architectures are evolving now?

What steps will make Event Processing an accepted standard subject in the teaching of Computer and Systems Science?

**precise and standardized definitions of basic concepts:
events, event relationships, event patterns, ...**

**languages, methods and algorithms form the
foundations of event processing,**

applications, applications, applications