AMIT – Active Middleware Technology

AMIT (and the active technology approach) vs. Other Tools

November 2001

IBM Research Laboratory in Haifa
Outline

- The active Paradigm
- Active vs. Passive
- AMIT vs. ECA
- AMIT – more
- Comparison to other tools
Passive applications

- The dominant paradigm is the passive one: request - reply
- Basic paradigm:
  - a request is performed
  - the current state is optionally evaluated to decide how should the request be performed
  - the action is taken, and reply is provided.
- Tools:
  - database queries
  - production rules (if... then...)
  - search engines...
  - ....
What is active behavior?

- An action which is performed without explicit request
- It reacts to some **detected transition** in the application domain (**event**)
- The timing may not be known in advance
- It can be a stand alone reaction, or (derived) part of a regular transaction.
Why is active behavior needed?

- In early days of programming languages it was used for interrupts/exception handling.
- Currently it becomes part of many application areas and used for:
  - monitor, control, resolve problems, alert, derive values, synchronize among systems...
  - originated in the "active database" area in the late 1980-ies
  - transferred to other areas in the second half of the 1990-ies (e.g. publish/subscribe)..
Active behavior with passive tools

- A possible approach is to use the same tools:
  - queries
  - production rules
  - etc...
But:

- **not efficient:**
  - need to run all of them periodically to check if condition for activation is satisfied.
  - these tools are state-based. Need to represent the transition in states, and some time the representation and processing is difficult [e.g. SQL/rule languages don't support temporal operators]...

- **not effective:**
  - too much time may pass before detection and action to be effective (e.g. arbitrage among two stock markets).
The first generation of active systems - the ECA paradigm

- Making the processing event driven
- The paradigm:
  - when event occurred
  - if a condition is satisfied
  - perform action
- In rule base systems: one more component was added to the rule
- In SQL: trigger is a limited form of this paradigm..
- Publish/subscribe with filtering is exactly ECA
- System management tools are ECA based...
Why ECA is not enough?

- The main problem with ECA is that applications need to react to an *application domain transition among states* while current tools react in practice to *computer-domain transition among states (= events)*.

- Example: *event*: quote of IBM stock given every 10 minutes. The occurrence of this event is a physical transition, however we need to react when IBM stock is up 3% during a time period of 2 hours....

- There is a semantic gap here, between needs and tools...
Situation to the rescue..

- The concept of *situation* is a transition in the "application/user domain" that the system should detect and react to...
- This is an abstraction over the universe of transitions in the same way that SQL query or view is an abstraction over the universe of states
- It is a natural building block in an active application
Using situations we can gain...

- **Effectiveness:**
  - Complex active applications become easy to express, thus more functionality may be enabled (e.g. making personalization feasible) and development time can be saved.

- **Efficiency:**
  - Less events flow on networks...
  - Less rules are activated..
  - Less code has to be maintained..
Two examples

1. If a "platinum customer" changed his/her stock portfolio at least twice this week in more than 10%, and his/her total investment is more than $1M, initiate a phone call to advise him/her.

2. If a "gold" or "platinum" customer deposited a sum of more than $10K in a checking account and did not withdraw money from the account within 2 days, initiate a phone call to advise him/her.
Implementation in passive style..

- Every day check all relevant clients.
- Needs to explicitly add to the database schema information about "major portfolio change" with dates...
- Needs to run query against the added values that represent transitions, and the state information to obtain status, investments...
- A lot of redundant computations...
- The state representation (database schema) becomes very complicated, because it needs additions... furthermore, the need to modify the database schema makes changes/additions difficult/impossible
Implementations through ECA

- Needs to refer only to customers with events of major portfolio change, thus makes it more efficient..

- But:
  - still need to maintain information in the database, because the first time that an event happens is not something we need to react to...
  - rules thrashing... (rules that create information for more rules and have no external impact)

- All ECA tools suffer from these deficiencies.
  - e.g. analysis of system management tools indicate that most rules are "internal", don't produce action outside the system databases..
Our vision for the next generation of active technology

- **Rationale:** IBM has an opportunity to be a market leader in this area, both as a "stand alone" concept, and as a value added embedded inside products and services.

- **Technical areas**
  - functionality: the SQL of the active world..
  - technical approach: connect to anything... integrate in existing GUI tools and toolkits.
  - adding intelligence: active behavior can be inferred in many cases.
  - generic active applications for domains..
Building the examples in our language

- **Basic events:**
  - buy-stock (customer, stock, value)
  - sell-stock (customer, stock, value)
  - deposit (customer, account, sum)
  - withdraw (customer, account, sum)

- **Implied events:**
  - major-portfolio-change (customer)

- **Context (life-spans):**
  - Major-portfolio-context (customer)
    - initiator: major-portfolio-context (account)
    - expires: one week
  - deposit-large-sum (customer)
    - initiator: deposit, sum > 10K
    - terminator: withdraw
    - expires: two days
Situation #1

- at least 2 major-portfolio-change
- key customer
- lifespan major-portfolio-context
Situation #2

- not withdrawal
- key account
- lifespan deposit-large-sum
Possible relationships to Major IBM Software products

- MQSI: Actions based on composite messages from possible different queues using the "situation manager" as an MQSI node.
- MQ-Workflow: Adding explicit control structures based on combination of workflow states, and external events.
- WAS: Adding tools to build personalized active web-based applications (e.g. push technology)
- DB2: situation oriented triggering.
- Tivoli: situation oriented rules as part of event correlation framework.
- WES: infrastructure for pervasive services
- Other products:
  - Lotus Sametime - groupware awareness rules
  - …
Amit Architecture

- Event adapter
- Event definitions
- Events
- Situation alerts
- Subscription and action manager

Components:
- System designers
- Authoring tool (Amit GUI)
- Event sources
- Users

Diagram shows the flow of information and interactions within the Amit Architecture.
Relationships to Rule based Technology

■ These are semi-orthogonal issues:
  ➔ rule based application may not be active
  ➔ active application may not need rules (just situation
    --- simple action).

■ The intersection:
  ➔ activated by a situation
  ➔ when the situation is detected, a deductive
    mechanism is applied (example: in our case study
    we identified when a client should be conducted.
    The consulting session may be handled by a
    deductive process, which can be dialogue driven)
In management and monitoring applications one of the main problems is the ability to deploy a relatively large set of rules over a relative small period of time.

These rules are “situation intensive” in the sense that the action typically is not a reaction to a single event.

The ability to use such “productivity tools” using high level abstractions may be a critical success factor in deployment.

Using Amit:
- Development is shorter. High abstractions lead to less rules relative to alternatives.
- Maintenance is cheaper: simplicity reduces maintenance cost.

We still need to verify this assumption relative to related products.
Relationship to Publish/Subscribe Technology

- Publish/subscribe is: event - action.
- Publish/subscribe with filtering is: event - condition - action.
- Intersection can lead to situation based publish/subscribe which can be more personalized, effective. (e.g. combination of Amit and Gryphon)
Relationship with Event Correlation Technology

- Event correlation technology (originated from network management) is aimed to filter out events that are related (e.g. to isolate the problem and filter out the symptoms).
- Event correlation has limited expressive power and it can handle a very specific pattern. (expressive power is typically reduced to: conjunction of events over a time window)
- From our point of view it is a type of application.
The full name: **Versata Studio & Logic Server** (former VisualAge Business Rules)

- **Declarative** rules rather than procedural, or: **What** rather than how

- Supported business rule types: Derivation (computational) rules, Validation rules, Presentation rules, Integrity rules, Constraints

- Versata translates system requirements into EJB, plain Java or HTML application or directly into relational database schema
**Versata Limitations:** Non-declarative requirements

Types- requirements that cannot be translated to declarative business rules:

- More complex relationships than parent-child such as siblings, cousins…
- Quantity-based discount schedules
- Batch driver loops (e.g. *Notify the contract administrator when a contract’s expiration data has passed, if the contract is of type “Service” and has a value of more than $10,000*)
- Workflow, including time-based and calendar-driven rules enforcement
- Data retrieval with a user-defined business function
Versata (cont.)

- Versata Limitations (more):
  - Does not support time
  - The tool requires “semi-programmer” skills
Blaze

- The full name: **Blaze Advisor (Brokat)**
- Supports simple ‘IF … THEN … ELSE’ rules.
- Used in the WebSphere Commerce Suite and in the Product Advisor, in ibm.com.
- Advisor Structured Rule Language (SRL) - natural, English-like language.
- Advisor rules can be written against true objects such as Java, CORBA or COM/ActiveX, but Advisor can have rules written against database rows mapped as “data-only” objects
Blaze Limitations (more):

- Does not support events
- No dynamic
- Decision tree - based
- Tightly coupled with object oriented programming
- DB oriented in principal
iSpheres

- The full name: iSpheres MetaApp Framework
- Provides tools that allow enterprises to build compositional information infrastructures with components both inside and outside the enterprise
- iSpheres employs compositional design that its primary advantage is that components expose what they do but not how they do it
- An iSpheres component exposes what it does in a simple way: it has input ports through which it receives messages and output ports through which it sends messages
Differs from our approach in: capabilities, scope and approach

Capability: The “event correlation” capabilities of iSpheres are **Syntactic**, roughly equivalent to a DBMS trigger, i.e. events are stored in a database, and whenever a new event satisfies an SQL query, it triggers some action.

In contrast, our active technology capabilities are **Semantic** and **Context Sensitive**, supporting higher-level abstractions, built-in support of temporal operators, context dependent operations.

Example: the situation *IBM STOCK HAS GONE UP in at least 3 percent twice within the same week and MICROSOFT stock has gone down in at least 1 percent during that period*.
iSpheres (cont.)

- **Scope:** iSpheres collects events from Applications using the meta-application wrapping
- **Our active technology Includes:**
  - Collecting events from different sources such as: sensors, devices, applications, databases, message queues etc., using event handlers
  - Event handlers for various distributed environments
  - Built-in dependency modeling capabilities that provides impact analysis, and support of proactive activities
iSpheres (cont.)

- **Approach:** iSpheres approach is to create “meta-application” wrapper for each type of application.
- **Nowadays it seems to be redundant,** given the fact that current standards such as WSDL, UDDI and SOAP serve the same purpose…
Event Correlation in Network Management

- Observation: a single problem can cause an event storm of symptom events
- Event correlation is a process that eliminates symptom events, and finds the problem out of these symptoms
The full name: VERITAS NerveCenter
A System network management tool
Correlates network events.
When a predefined network condition is detected, NerveCenter stores the event information in a finite state machine called an *alarm*.
The *alarm* continues to track the status of the object being monitored.
To correlate and filter this data, VERITAS relies on configurable models of network and system behavior, called behavior models, for each type of managed resource.
The full name: SMARTS InCharge
A System network management tool
Correlates events by employing a coding technique that matches alarms with signatures of known problems in real-time
A set of events that represent symptoms of problems is treated as a code that identifies the problem.
A codebook is a set of events that must be monitored to distinguish the problems of interest from each other.
The supported pattern on event history is a conjunction of events within a time window
Incharge Capabilities

- Accepts as input causality graph among events, and classification to “symptoms”, “problem”, or none
- Reduces the graph by eliminating nodes that are not either problems, or symptoms related directly to problems
- Code the relationships among symptoms and problems in a way that enable the detection of problems
The Coding Method

- Each problem is represented using a sequence of bits, each bit represents a symptom.
- Example: There are three symptoms, problem 1 propagates to the first and third symptoms, its representation is ‘101’.
## Correlation matrix

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Incharge limitations

• Weak sense of causality. The semantics of causality is event B follows event A after no more than a time constant T in frequency > F.

• In reality it is possible that:
  – There is a complex causality (if A and B then C; if A occurs and B does not occur then C; If A and B occur simultaneously then C…)
  – All these patterns can be expressed in Amit and not in Incharge.
Incharge limitations (cont.)

• Furthermore:
  – Events can be correlated only within a certain CONTEXT
  – The temporal causality can be of different time-frames for different correlations
  – The causality may be of multiple symptoms to multiple problems
  – The level of correlation may be stochastic….

Amit supports all these variations, Incharge – none
Research Prototypes

- **ODE - AT&T Bell Laboratories**
  - Limited to database events only
  - Detects composite events over an event history that contains all event occurrences.
  - This information can only be used to impose some filtering conditions (masks) and equality conditions (parameters) on events that participate in an event expression (composite event).
Research Prototypes (cont.)

- **Snoop - University of Florida**
  - Supports both database event and external events
  - Limited expressive capabilities for the definition of time internals using the operators A, A*, P, and P* in association with a parameter context
  - Snoops cannot express all possibilities of event reuse (consumption) policies
  - Although semantic information is reported with events in Snoop, this information cannot be used during the event composition
After learning all the related products we still assert the same basic assumption:

- In management and monitoring applications one of the main problems is the ability to deploy a relatively large set of rules over a relative small period of time.
- These rules are “situation intensive” in the sense that the action typically is not a reaction to a single event.
- The ability to use such “productivity tools” using high level abstractions may be a critical success factor in deployment.
- Using Amit:
  - Development is shorter. High abstractions lead to less rules relative to alternatives.
  - Maintenance is cheaper: simplicity reduces maintenance cost.
Conclusion (still emerging...)

- Active technology is an emerging domain...
- We have set a vision and a set of related components (Amit, Adi, Arad)
- Has some relations with other technologies (rule base, pub/sub..), but stands on its own feet..
- Need to position:
  - within IBM products (exposed or embedded)
  - within IBM services
  - composition architecture
- There is a potential for IBM to be market leader and a limited time window to realize it!