Event-Driven Architectures for Distributed Crisis Management

Caltech Infospheres Group
Professor K. Mani Chandy,
Brian Aydemir, Elliott Karpilovsky,
Dr. Dan Zimmerman
What is a Crisis?

- Usually are uncontrollable, unpredictable, and have dire consequences
  - Airplane crashes, power outages, wildfires, etc.
- Require quick, decisive action
- Require groups of people to work together
  - Police, fire departments, news teams, etc.
  - Distributed due to control issues
Dealing With Crises

- Three primary problems arise
  - Acquiring data
  - Analyzing data
  - Executing actions

- Normal management schemes often are inadequate
  - Rules engines, enterprise computing systems, etc.

- Abstract and concrete models proposed to cover these crises
Abstract Model: Variables

- **External variables** represent real-world quantities
- **Internal variables** are the results of computations concerning other variables
- Special *time* variable reflects progress of time in real world
Abstract Model: Execution

- *When-then rules* sole means of specifying system behavior
  - *When* history-predicate *then* state-change-specification
- When new information becomes available, all when-then rules executed immediately and concurrently
- Cannot be implemented
- Useful as a starting point for a concrete model
Concrete Model Overview

- Three components comprise system: sensors, actuators, and event processors
  - Sensors generate events based on external properties of the monitored system
  - Actuators consume events and change external properties
  - Event processors consume events and generate new ones
- New information entering the system is represented by events
The Role of Sensors

- Send information into the network
- All information specified as attribute-value pairs
  - (subject, earthquake)
  - (time, 9:15:35)
  - (intensity, 3.0)
  - (latitude, 34.057°N)
  - (longitude, 118.834°W)
The Role of Actuators

- Specify reactions to information
- Reaction can notify other software or individuals
  - Can cause change in the real world
- Must be activated via events
  - Cell phones, pagers, sirens, etc.
The Role of Event Processors

- Event processors implement the internal variables and *when* part of when-then rules
- Operate only over local variables
- Specify a set of *input ports*, a *state transition function*, an *event generation function*, and a *timeout* value
Event Processor Specification

- State transition function updates local variables
- Event generation function creates new events
- Timeout acts as an event if none received in specified time
  - Can be changed by state transition function
- Set of input ports dictates information to process
  - Can be prioritized
Example: Blood Supply

- Monitor blood supplies throughout L.A. County
- Event processor for health director implements when-then rule
  - **when**
    - one-week moving-point average blood plasma usage in L.A. County exceeds three-quarters of one-week moving-point average blood plasma inventory available in L.A. County,
  - **then**
    - send alert
Example: Blood Supply

- Event processor for health director:
  - Has two ports: `blood_available` and `blood_used`
  - Local variables `blood_available[]`, `blood_used[]`, `avg_blood_available` and `avg_blood_used`
  - State transition function appropriately modifies local variables based on new information
  - Event generation function sends out alert to director when critical condition satisfied
  - `Timeout` is 24hrs
Example: Blood Supply

Hospitals

Blood Banks

Health Director
Example: Blood Supply

Hospitals

Blood Banks

Health Director

Event Processor

blood_used

blood_available

avg_blood_used:

avg_blood_available:

blood_used[]

blood_available[]

Caltech Infospheres Group
http://www.infospheres.caltech.edu/

Event-Driven Architectures for Dist. Crisis Mgmt.
3 November 2003
Example: Blood Supply

- Hospitals
- Blood Banks
  - Blood available
  - Blood used
- Health Director
- Event Processor
  - avg_blood_used:
  - avg_blood_available:
  - blood_used[]
  - blood_available[]

Example:
- Blood Supply

Event-Driven Architectures for Dist. Crisis Mgmt.
3 November 2003
Example: Blood Supply

Hospitals

(blood used)

Blood Banks

(blood available)

blood_used

blood_available

avg_blood_used: 9,100 L
avg_blood_available: 12,000 L
blood_used[]
blood_available[]

Health Director

(alert)

Event Processor
Implementation

- Messages implemented in XML
- States for processors implemented in XML, transitions implemented in XSLT
- System implemented in Java
- Persistent storage used to catalog information
Further Work

- Increase system robustness and fault tolerance
- Address security issues
- Create simpler interface for use by non-technical personnel
Conclusions

- *When-then rules* provide a simple way to help emergency response teams deal with dynamic, complex crises.

- When-then rules can be adequately approximated by *event processors*.

- First step towards a truly generic crisis management system.
Questions?